LEARNING MULTI-UAV POLICIES USING DEEP REINFORCEMENT LEARNING FOR FLOOD AREA COVERAGE AND OBJECT TRACKING

DOCTORAL THESIS

BY

ARMAAN GARG
(2019CSZ0002)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROPAR
RUPNAGAR, 140001, PUNJAB, INDIA

APRIL, 2024

Learning Multi-UAV Policies Using Deep Reinforcement Learning for Flood Area Coverage and Object Tracking

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

by

Armaan Garg

(2019CSZ0002)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROPAR

April, 2024

Dedicated to
My beloved Parents,
My sweet Wife
and
My charming Brother

It is from their love, care and support, I drive my strengths...

Declaration of Originality

I hereby declare that the work which is being presented in the thesis entitled **Learning Multi-UAV Policies Using Deep Reinforcement Learning for Flood Area Coverage and Object Tracking** has been solely authored by me. It presents the result of my own independent investigation/research conducted during the time period from July 2019 to November 2023 under the supervision of Dr. Shashi Shekhar Jha, Assistant Professor in the Department of Computer Science & Engineering at the Indian Institute of Technology Ropar. To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted or accepted elsewhere, in part or in full, for the award of any degree, diploma, fellowship, associateship, or similar title of any university or institution. Further, due credit has been attributed to the relevant state-of-the-art and collaborations (if any) with appropriate citations and acknowledgements, in line with established ethical norms and practices. I also declare that any idea/data/fact/source stated in my thesis has not been fabricated/ falsified/ misrepresented. All the principles of academic honesty and integrity have been followed. I fully understand that if the thesis is found to be unoriginal, fabricated, or plagiarized, the Institute reserves the right to withdraw the thesis from its archive and revoke the associated Degree conferred. Additionally, the Institute also reserves the right to appraise all concerned sections of society of the matter for their information and necessary action (if any). If accepted, I hereby consent for my thesis to be available online in the Institute's Open Access repository, inter-library loan, and the title & abstract to be made available to outside organizations.

[illegible]
Signature

Name: Armaan Garg
Entry Number: 2019CSZ0002
Program: PhD
Department: Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar, Punjab 140001

Date: 01/04/24

Acknowledgement

As I complete this journey towards obtaining my PhD, I am filled with gratitude and humbled by the support and guidance of those who have helped me along the way. It is my pleasure to take this opportunity to express my heartfelt thanks to the following people.

First and foremost, I would like to extend my sincerest gratitude to my thesis supervisor, Dr. Shashi Shekhar Jha, for his unwavering support, encouragement, and guidance throughout my research. Your expertise and your dedication to my academic and personal growth have been invaluable. I am grateful for the opportunities you provided me to grow as a scholar and for the trust you placed in me.

I would also like to express my gratitude to the members of my Doctoral committee, Dr. Sujata Pal, Dr. Shweta Jain, Dr. Nitin Auluck and Dr. Reet Kamal Tiwari for their insightful comments, constructive criticism, and valuable suggestions. Their guidance and expertise have been instrumental in shaping the direction and outcome of my research.

I am also grateful to my colleagues and friends who have provided me with emotional support and encouragement throughout this process. Your kindness and friendship have meant the world to me.

I would like to acknowledge the fellowship support provided by TCS Research, without which this research would not have been possible.

Finally, I would like to thank my family, especially my wife, my parents and my elder brother for their unwavering love, support, and encouragement. Their sacrifices have been a constant source of motivation and inspiration, and I am grateful for the opportunities they have provided me.

Thank you to everyone who has made this journey possible. I am forever grateful for your support.

Sincerely,
Armaan Garg

Certificate

This is to certify that the thesis entitled **Learning Multi-UAV Policies Using Deep Reinforcement Learning for Flood Area Coverage and Object Tracking**, submitted by **Armaan Garg** for the award of the degree of **Doctor of Philosophy** of Indian Institute of Technology Ropar, is a record of bonafide research work carried out under my guidance and supervision. To the best of my knowledge and belief, the work presented in this thesis is original and has not been submitted, either in part or full, for the award of any other degree, diploma, fellowship, associateship or similar title of any university or institution.

In my (our) opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the Degree.

[illegible]

Signature of the Supervisor(s)
Dr. Shashi Shekhar Jha
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar, Punjab 140001
Date: 01/04/24

Lay Summary

The disaster relief operations during floods require real-time information of the ground situation from the flooded area. Finding critical regions in flood affected area in a limited time frame is crucial for effective relief planning. With the advent of technology, Unmanned Aerial Vehicles (UAVs) are being deployed for active flood monitoring and area coverage to provide support during search-and-rescue operations. Consequently, UAVs extend the capabilities of the rescue teams in remote sensing and terrain monitoring tasks. However, majority of the current UAV-based deployments rely on expert pilots for remotely flying the UAVs and performing the desired task(s). This dependency on human pilots limit the UAV's operability in unknown and highly dynamic environments as it requires the pilot to control the UAV(s) with limited line-of-sight range. Hence, there needs to be correct autonomy in place for the team of UAVs to perform desirable data-gathering tasks in unknown and unseen environments. However, it is not trivial to make UAVs fly autonomously to execute complex tasks in critical environments, such as floods.

Recent advancements in robot control algorithms have made it possible to achieve autonomous UAV flight. This has been achieved using algorithms like model-based and heuristic approaches. However, applying these methods in dynamic environments is challenging due to limited knowledge of the surroundings. Reinforcement learning (RL) algorithms provide a framework by enabling UAVs to learn appropriate control sequences through interactions with the environment, making it capable for autonomous flights in dynamic environments. However, standard RL algorithms face multiple difficulties, requiring a multi-UAV team to randomly explore before finding an effective strategy to perform tasks such as cooperative area coverage under the constraint of limited battery. In this thesis, Deep RL algorithms are proposed with novel exploration strategies for generating autonomous controls for a multi-UAV system to perform tasks such as flood area coverage, real-time path planning and object tracking.

The first objective is to identify critical regions during floods using UAVs in a cooperative manner, aiming to maximize area coverage within a limited time-frame. The algorithms proposed for this task can further be classified into solutions for discrete and continuous action spaces. Additionally, decentralized trained multi-UAV policies are proposed in contrast to centrally trained policies, enhancing their applicability. Subsequently, after identifying critical regions, the thesis addresses the problem of identifying serviceable paths for evacuation vehicles using autonomous UAVs to assist rescue operations. Furthermore, this thesis also addresses the problem of tracking a moving convoy of vehicles using multiple UAVs to enhance coverage for security and surveillance purposes.

The proposed solutions are discussed as individual chapters in this thesis, each addressing a specific objective. Following the Introduction and Literature review chapters, Chapter 3, "Directed Explorations During Flood Disasters Using Multi-UAV System," focuses on strategies for identifying critical flood areas. Moving forward, Chapter 4, "Continuous Multi-UAV Control with Directed Explorations during Floods," focuses on learning continuous multi-UAV policies to enhance control strategies in flood scenarios.

Chapter 5, “Autonomous Flood Area Coverage using Decentralized Multi-UAV System,” discusses the algorithm for achieving autonomous multi-UAV controls using decentralized training. Shifting focus to Chapter 6, “Real-Time Serviceable Path Planning during Floods,” addresses the problem of real-time path planning using UAVs, specifically for evacuation vehicle navigation during floods. Extending the application scope, Chapter 7, “Autonomous Multi-UAV Control for Moving Convoy Tracking,” aims to improve target function approximation for the Deep Deterministic Policy Gradient algorithm to track a moving convoy using multiple autonomous UAVs. Each chapter contributes uniquely to enhance UAV capabilities for autonomous control in distinct scenarios. Finally, Chapter 8 concludes this thesis.

Abstract

During disasters, such as floods, it is crucial to get real-time ground information for planning rescue and response operations. With the advent of Unmanned Aerial Vehicles (UAVs), flood monitoring capabilities have improved significantly, yet the dependency on expert human pilots limit their operational scalability in unknown flood-like environments. To tackle such issues, autonomous multi-UAV systems can be deployed to perform the task of cooperative flood area coverage without human intervention. Recent advances in robot control algorithms have attempted to deploy autonomous UAV systems for various tasks, particularly, leveraging deep reinforcement learning (Deep RL) algorithms. However, training Deep RL algorithms pose various challenges, such as sparse early rewards, target approximation errors, and overestimation bias. These limitations often leads to sub-optimal policies, especially when learning complex value functions in dynamic and stochastic environments, like floods.

In this thesis, the focus is on learning effective autonomous multi-UAV policies for flood area coverage, path planning, and object tracking by introducing novel exploration strategies and target function approximators. The proposed solutions aim to mitigate the limitations associated with training Deep RL policies for multi-UAV systems in complex environments, such as floods. Domain knowledge based directed explorations are introduced using water-flow algorithms, viz., D8 and D-infinity to expedite training of Deep RL policies and to accumulate high rewards especially in initial episodes. The proposed algorithms, D8QL (for discrete state-space) and D8DQN (for continuous state-space) uses an ϵ_1-ϵ_2 exploration strategy, distinguishing them from purely random exploration based ϵ-*greedy* strategy. Further, D3S algorithm is presented to deal with continuous action-spaces for smoother UAV motion. Additionally, a decentralized training paradigm is introduced to learn multi-UAV policies, as opposed to centrally trained policies, to perform flood area coverage and to identify critical regions. The decentralized approach enables flexible response capabilities in scenarios where communication with the ground control station might be restricted or limited. Further, to mitigate poor training due to random initialization of target networks in Deep RL based actor-citric algorithms, a Gaussian process regression (GPR) based value function approximation technique is proposed. GPR is used as the target critic to improve the multi-UAV policy to track a convoy of moving vehicles. This thesis also presents a multi-UAV path planning strategy to navigate waterborne evacuation vehicles to reach critical location(s) during floods. A minimum node expansion strategy is proposed to tackle the issue of exponential complexity associated with the A* algorithm in large state-space environments.

All the proposed algorithms are benchmarked against established Deep RL baselines and state-of-the-art algorithms from the recent literature. The results show that the proposed algorithms outperform other techniques across multiple performance measures. The proposed algorithms provide improved autonomous solutions for multi-UAV operations in flood relief tasks, offering critical area coverage, efficient path planning, and continuous object tracking.

Keywords: Unmanned Aerial Vehicles, Multi-UAV systems, Deep Reinforcement Learning, Joint Area Coverage, Flood Disaster Response, Real-Time Path Planning, Multi-Object Tracking, Gaussian Process Regression.

List of Publications

Journals

1. **A. Garg** and S. S. Jha, “Deep Deterministic Policy Gradient based Multi-UAV Control for Moving Convoy Tracking,” Engineering Applications of Artificial Intelligence, vol. 126, p. 107099, 2023, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.107099.

2. **A. Garg** and S. S. Jha, “On Learning Continuous Multi-UAV Controls with Directed Explorations for Flood Area Coverage,” Robotics and Autonomous Systems, 2023 — Under review.

3. **A. Garg** and S. S. Jha, “Autonomous Multi-UAV Control for Disaster Response with Reinforcement Learning - An Overview,” Annual Reviews in Control, 2024 — Under review.

4. **A. Garg** and S. S. Jha, “Multi-UAV Assisted Flood Navigation of Waterborne Vehicles using Deep Reinforcement Learning,” ASME Journal of Computing and Information Science in Engineering, 2024 [Invited Paper] — Under Review.

Conferences & Workshops

1. **A. Garg** and S. S. Jha, “Autonomous Flood Area Coverage using Decentralized Multi-UAV System with Directed Explorations,” in Adaptive and Learning Agents (ALA) Workshop at the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2023.

2. **A. Garg** and S. S. Jha, “Real-time Flood Navigation: Multi-UAV Assisted Vehicle Guidance via Deep Reinforcement Learning,” in Learning Robot Super Autonomy Workshop at the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023 - Accepted.

3. **A. Garg** and S. S. Jha, “Decentralized Critical Area Coverage using Multi-UAV System with Guided Explorations during Floods,” in 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), 2023, pp. 1–6. DOI: 10.1109/CASE56687.2023.10260489.

4. **A. Garg** and S. S. Jha, “Real-Time Serviceable Path Planning using UAVs for Waterborne Vehicle Navigation during Floods,” Advances In Robotics - 6th International Conference of The Robotics Society, 2023. DOI: 10.1145/3610419.3610433 - (Invited for journal submission to ASME).

5. **A. Garg** and S. S. Jha, “Directed Explorations during Flood Disasters using Multi-UAV System,” in 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), 2022, pp. 2154–2161. DOI: 10.1109/CASE49997.2022.9926454.

6. P. Kaushik, **A. Garg**, and S. S. Jha, "On Learning Multi-UAV Policy for Multi-Object Tracking and Formation Control," in 2021 IEEE 18th India Council International Conference (INDICON), 2021, pp. 1–6. DOI: 10.1109/INDICON52576.2021.9691567.

Contents

List of Figures

List of Algorithms

List of Symbols

U	Set of UAVs
u_i	i^{th} UAV
F	Set of UAVs' FoVs (field of view)
v_i	FoV of i^{th} UAV
n	Number of UAVs
$\mathbb{H}$	Altitude of UAV(s)
$\psi_t^{u_i}$	UAV energy at time t
S	Set of states
s^{u_i}	State of i^{th} UAV
O	Set of observations
$o_t^{u_i}$	Observation of i^{th} UAV at time t
A	Set of actions
a^{u_i}	Action of i^{th} UAV
$P_{ss'}$	State transition function
$R_t^{u_i}$	Function which returns the reward received by UAV u_i at time t
c	Environment location (grid cell) that is being observed by a UAV
$\mathbb{P}_c^{u_i}$	Population density at location c that is currently being observed by UAV u_i
$e_c^{u_i}$	Terrain elevation at location c that is currently being observed by UAV u_i
h_{c_t}	Water level/depth at location c at time t
f_{rate}	Water flow rate
$\mathbb{Z}_c$	Critical level of location c
I_c	Information gain from location c
ω_t^{in}	Inward water discharge for location c at time t
ω_t^{out}	Outward water discharge for location c at time t
ϕ	Manning roughness coefficient

g	Value of gravity
s_g	Surface slope
s_f	Friction slope
d	Function which return the distance between two environment locations/cells or the 2D projected positions of two UAVs
Ω	Transmission power of the UAV
f_c	Carrier frequency
$\mathcal{V}$	UAV motion speed
ϵ	Exploration probability
γ	Discount factor
π	Policy
Q	Function which returns the state-action value (i.e., expected return of when starting in state s, taking action a, and following policy π thereafter)
θ_1, θ_2	UAV ventral camera half angles
$\mathcal{Z}$	Experience replay buffer
$\mathcal{B}$	Sampled mini-batch from the experience replay buffer
θ^Q	Parameters of learning critic network
μ'	Target actor network
$\theta^{\mu'}$	Parameters of target actor network
T	Set of target vehicles
τ_i	i^{th} target vehicle
$\mathbb{T}$	Function which returns the incentive corresponding to trajectory (UAV-to-vehicle) alignment
$\mathbb{I}$	Function which returns the incentive corresponding to overlap (UAV-to-UAV)

List of Abbreviations

Terms	Abbreviations
UAV	Unmanned Aerial Vehicle
FoV	Field of View
WBV	Waterborne Vehicle
GCU	Ground Control Unit
RL	Reinforcement Learning
Deep RL	Deep Reinforcement Learning
MARL	Multi-agent Reinforcement Learning
MDP	Markov Decision Process
DEM	Digital Elevation Model
DQN	Deep Q-Network
D8DQN	D8 exploration in DQN
D8QL	D8 exploration in Q-Learning
DDPG	Deep Deterministic Policy Gradient
MADDPG	Multi-agent Deep Deterministic Policy Gradient
GPR-MADDPG	Gaussian Process regression for target value function approximation in MADDPG
MA-A2C	Multi-agent Advanced Actor-Critic
MA-A3C	Multi-agent Asynchronous Advanced Actor-Critic
TD3	Twin Delayed Deep Deterministic Policy Gradient
RW	Random Walk
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
RRT	Rapidly Exploring Random Tree
MEA*	Minimum Expansion A*

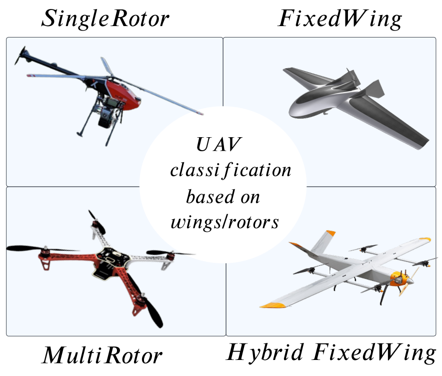
1| Introduction

Floods are one of the most critical and frequently occurring calamities across different parts of the world. In the decade spanning from 2010 to 2019, the global monetary losses attributed to floods are estimated to have averaged approximately 30 billion USD annually [2, 3]. During this period, floods affected millions of people, causing over 6,000 deaths each year and rendering more than 0.28 million people homeless [3]. Flood relief planning is a critical task where real-time information is required at regular intervals to carry out rescue operations effectively. With the advent of technology, Unmanned Aerial Vehicles (UAVs) are being deployed for active flood monitoring [4] and area coverage [5, 6] to provide support during search-and-rescue operations.

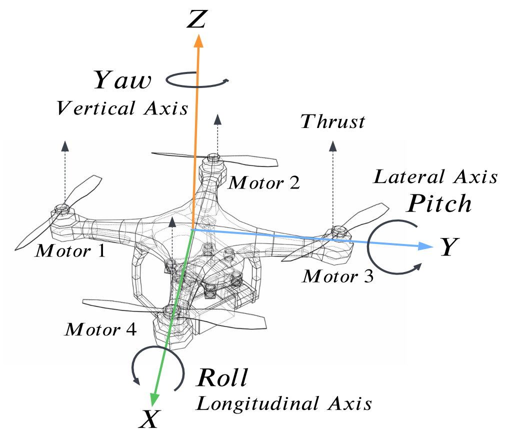
UAVs further extend the capabilities of the rescue teams in remote sensing and terrain monitoring tasks [4, 7]. However, the majority of the current UAV-based deployments rely on expert pilots [8, 9, 10] for providing command and control to perform the desired task(s). This dependency limits the UAV's operability in unknown and highly dynamic environments as it requires the pilot to provide the UAV with appropriate actions after perceiving the observed information. Therefore, its important to have correct autonomy in place for a team of UAVs to perform desirable data-gathering tasks in unknown and unseen environments. However, evolving autonomous multi-UAV policies in order to gather data from critical regions of the flood-affected areas is not trivial.

Learning autonomous policies for UAVs has proven to be difficult because of factors such as wind conditions, precipitation, static and dynamic obstacles, changing regulatory restrictions, etc., making it difficult to establish pre-defined control protocols. In dynamic and stochastic environments, the ability to adapt in real-time is crucial. Fixed control strategies prove ineffective in such scenarios, necessitating the development of flexible approaches for drones to cope with ever-changing conditions [11]. To enhance UAV robustness, the incorporation of interactive algorithms (i.e., learning from trial-and-error), particularly Reinforcement Learning (RL), has become a focal area of research [12].

Recent advancements in RL, including the application of Deep Reinforcement Learning (Deep RL) [13], have further extended the capabilities of UAVs [14]. Deep RL has enabled the UAVs to comprehend complex state-action functions, allowing them to navigate intricate environments [15] with high degree of precision and intelligence. This trail-and-error approach with continuous interactions with the environment empowers UAVs to autonomously learn and optimize actions based on experiences and rewards. Reinforcement learning sets itself apart from other approaches by not necessitating labeled input/output pairs or explicit corrections for sub-optimal actions. Instead, it focuses on finding a balance between exploration (of unknown environment) and exploitation (of current experience). This thesis explores the applications of RL, particularly Deep RL, for multi-UAV control. Delving into its framework, the focus is on model-free algorithms as opposed to model-based algorithms that infer the model of the environment from its



(a)



(b)

Figure 1.1: (a) Types of UAVs classified based on wings and rotors and (b) Its rotations across different axis.

observations and then plan a solution using that model. For flood response applications, the model-free Deep RL algorithms are deemed to be more suitable, enabling UAVs to dynamically adjust and make instantaneous decisions. This adaptability is crucial in addressing the inherent uncertainty and variability of flood conditions.
In the following sub-sections an overview of different types of UAVs and their characteristics will be presented. Furthermore, a concise overview of Reinforcement Learning will also be provided. Additionally, emphasis will be placed on highlighting the primary research challenges in learning RL policies, specifically focusing on multi-UAV policies for flood disaster response applications. To conclude this section, a summary will be provided outlining the research objectives and the structure of the thesis.

1.1 Unmanned Aerial Vehicles (UAVs)

A UAV, commonly known as a drone, is an aircraft without a pilot onboard. As can be seen in Figure 1.1a, UAVs come in various types, broadly categorized as multi-rotor, fixed-wing, single-rotor and hybrid, each tailored to specific applications. Multi-rotor drones demonstrate agility in tasks like aerial inspection, while fixed-wing drones excel in long-range mapping and surveillance. Hybrid drones handle heavy payloads, and single-rotor helicopters are ideal for applications where extended hovering is required. Each type has distinct advantages and trade-offs in maneuverability, endurance and cost. A UAV is controlled either remotely by a pilot commanding through the ground control station or by autonomous control commands generated using computer algorithms. As can be seen in Figure 1.1b, controlling the UAV movement involves the following actions [16]:

1. **Throttle:** This action regulates the propeller's motor power due to which an increase in throttle increases the speed and altitude of the UAV. It determines the

amount of thrust generated by the propulsion system.

2. **Pitch:** This action helps in the forward and backward movement of the UAV. At hover/rest position (considered pitch = 0), the UAV alignment is parallel to the ground. A forward push in reference to the lateral axis tilts the UAV head downwards as the tail goes up resulting in a forward movement. Similarly, a backward push tilts the UAV head upwards as the tail goes down resulting in a backward movement of the UAV.

3. **Roll:** This action is responsible for the UAV's bank or tilt from side to side, which affects its turning ability. Rolling to the left causes the left side of the UAV's body (in reference to the UAV head) to go down bringing the right side upwards and tilting the UAV in the left direction. Similarly, rolling towards the right causes the right side of the UAV's body to go down bringing the left side upwards and tilting the UAV in the right direction.

4. **Yaw:** Yaw controls the UAV's rotation around its vertical axis, which affects its orientation and heading. A push in the left direction shifts the UAV head anticlockwise whereas a push in the right direction shifts the UAV head clockwise.

1.2 Multi-UAV Systems

A Multi-UAV system [17], refers to a group of Unmanned Aerial Vehicles (UAVs) working collaboratively towards a global objective. These UAVs work as a team, sharing information and coordinating their actions in real-time to accomplish the objective. For tasks like flood area coverage, employing a cooperative approach enables UAVs to cover larger regions and accomplish complex tasks more efficiently than a single UAV could manage alone. Autonomous policies for multi-UAV systems can be learnt using a centralized or decentralized training methodology. In a centralized setup [18], a ground control unit (GCU) collects global state information and coordinates the actions of all UAVs. This approach often involves a joint policy that guides the entire fleet, making decisions based on a holistic understanding of the environment and task requirements.
Conversely, training decentralized systems allows UAVs to learn policies based on local experiences. Each UAV learns a local policy while sharing information with neighboring UAVs. Decentralized systems often exhibit robustness and adaptability, as they can continue functioning even if individual UAVs within the network are disrupted. However, policies trained in a decentralized manner tend to converge slowly and frequently exhibit challenges in coordination, often resulting in overlapped exploration when compared to centrally trained systems.
This thesis introduces domain-specific solutions to address the limitations of standard Deep RL algorithms with randomly initialized target functions, particularly in tasks with sparse early rewards, such as flood area coverage and object tracking. The proposed Deep RL algorithms cater to both discrete and continuous action settings of UAVs. Furthermore,

a decentralized algorithm is proposed for training multi-UAV policies, in contract to the centralized approach. Expanding on the application scope beyond flood area coverage, this thesis also explores multi-UAV applications in path planning and target tracking.

Before delving into the detailed overview of the proposed research objectives, a brief discussion about RL (Reinforcement Learning) and Deep RL (Deep Reinforcement Learning) will be presented in the subsequent subsections.

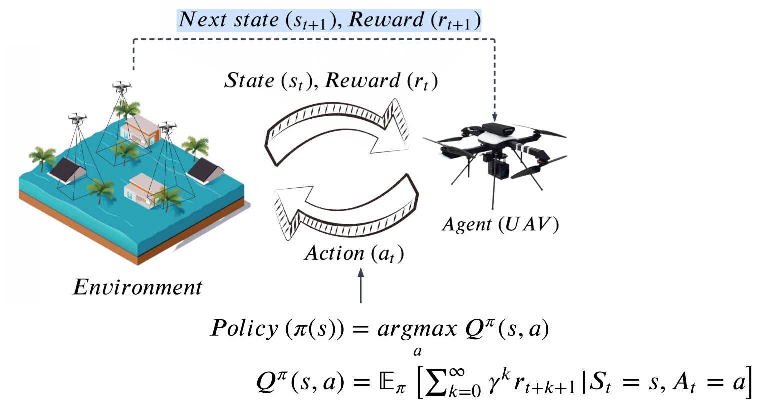

$Next\ state\ (s_{t+1}),\ Reward\ (r_{t+1})$
$State\ (s_t),\ Reward\ (r_t)$
$Agent\ (UAV)$
$Action\ (a_t)$
$Environment$
$Policy\ (\pi(s)) = \underset{a}{argmax}\ Q^{\pi}(s,a)$
$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a\right]$

Figure 1.2: The UAV–and-environment interaction in a Markov decision process.

1.3 Reinforcement Learning (RL)

RL algorithms [19] provide a general framework that helps in learning autonomous policies for UAVs by interacting with the environment in a trial-and-error fashion, as seen in Figure 1.2. RL algorithms aim at learning the optimal behaviour for an agent in an environment described in the form of a Markov decision process (MDP). An MDP is a decision process that provides the mathematical framework for modelling decision-making in scenarios involving partially random and partially controlled outcomes. Its important to note that the state transitions within an MDP adhere to the Markov property [19]. RL uses the formal framework of MDPs to define the interaction between a learning agent (i.e., a UAV in the given scenario) and its environment in terms of states, actions, and rewards [19]. In the given context, the MDP can be formally presented as a quintuple $< S, A, P_{ss'}, R, s_0 >$ where S represents the state of the UAV, A denotes its feasible action set, $P_{ss'}$ represents the state transition function from state s to s', R denotes the reward function and s_0 denotes the starting configuration of the UAV in the environment. A state-action value (Q-value) function is defined as the cumulative rewards accumulated by the UAV(s) based on its action in a given state. The idea revolves around iteratively interacting with the environment to determine the best action for a given state. This process balances exploration (discovering new possibilities) and exploitation (capitalizing on experience), ultimately leading to an optimal state-action value function.

The state-action value function is given as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1}\right], s \in S, a \in A \tag{1.1}$$

where t denotes time, π represents the current policy and r denotes the rewards/incentives. $0 \leq \gamma < 1$ is the discount factor to solve the infinite horizon problem (i.e., infinite reward calculation as the number of time steps tends to infinity). Hence, a policy can be defined as a mapping from the state space to a feasible action set. Various update rules are established in different reinforcement learning algorithms to enhance the policy, aiming to maximize the total rewards accumulated over time by adjusting the state-action value function $Q^{\pi}(s_t, a_t)$ towards the optimal value function $Q^{*}(s_t, a_t)$.

RL algorithms can be broadly categorized into model-based and model-free algorithms. Model-based RL algorithms rely on the model of the environment ($P_{ss'}$) to train a policy. These algorithms [20] are computationally more expensive as compared to model-free algorithms. However, they often learn better value function estimates with fewer environmental interactions and provides better interim performance as compared to model-free algorithms. A few examples of model-based algorithms are Dyna style algorithms [21], Imagination-Augmented Agents (I2A) [22], AlphaZero [23], Dynamic programming [21] among others. In contrast, being computationally less expensive, model-free algorithms can accommodate a comparatively larger state-action space given the same computational resources. Additionally, in dynamic and stochastic environments like floods, model-free algorithms are considered better suited than model-based algorithms. This is because model-free methods learn directly from environmental interactions, adapting without needing a detailed model of the environment, i.e., they do not need an explicit transition function $P_{ss'}$ [24]. The subsequent discussions are centered on model-free algorithms due to their suitability and adaptability in operating effectively within stochastic and uncertain environments, such as those encountered in scenarios like flood area coverage, target tracking, etc.

1.4 Deep Reinforcement Learning (Deep RL)

Deep RL combines deep learning and reinforcement learning to learn optimal strategies by approximating the value function. The integration of deep neural networks (DNN) has significantly advanced the capabilities of RL algorithms to learn complex value functions for autonomous controls in various settings, including multi-agent systems. Their deep architectures enable understanding of high-dimensional data, facilitating informed decisions and adaptive responses to real-world problems. Deep RL techniques, such as Deep Q-Network (DQN) [13], Double DQN [25], Dueling DQN [26], and others [27], have achieved super human performance, revolutionizing a wide array of applications. These advancements have significantly influenced various domains, from immersive applications such as gaming to autonomous systems and robotics. Along with deep networks these algorithms also use experience replay, and prioritized sampling to learn complex policy functions in discrete action spaces. In a specific application discussed in [28], DQN is employed to optimize the actions of multiple UAVs for flood area monitoring, focusing on planning UAV trajectories, including bank angles for fixed-wing UAVs, in a decentralized flood monitoring approach. In another study [29], a Double DQN algorithm is employed to maximize UAV coverage while efficiently managing power constraints. The proposed model considers various landing positions and navigates no-fly zones, utilizing spatial maps as input for training convolutional network layers. These studies merely scratch the surface of the profound impact these algorithms can have on addressing varied real-world challenges. Further, algorithms like Advantage Actor-Critic (A2C) [30, 31], Asynchronous Advantage Actor-Critic (A3C) [30], Deep Deterministic Policy Gradient (DDPG) [32], Soft Actor-Critic (SAC) [33], and Twin Delayed Deep Deterministic Policy Gradient (TD3) [34] have significantly broadened the scope of control possibilities in continuous action spaces [35, 36, 37].

However, training Deep RL policies, especially in complex environments like floods, poses intricate challenges. In the following sub-sections, this thesis will delve into these challenges and elaborate on the potential solutions, which constitute the primary contributions of the proposed research works.

1.5 Challenges in Learning RL policies

It is important to highlight that RL policies may not always converge, particularly in intricate environments [34, 38]. This is because, in complex and stochastic environments, standard RL algorithms often struggle to accumulate rewards during the early stages of training, resulting in sub-optimal control policies [34, 39]. Also, high fluctuation in accumulated rewards from episode to episode can lead to unstable learning. Additionally, Deep RL algorithms usually make use of fixed target networks to stabilize learning. However, as these target network functions are randomly initialized they lead to delayed convergence and require a substantial number of samples [40]. The poorly defined target functions also introduce overestimation bias in the policy, causing the policy to diverge [34]. To address the problem of overestimation bias (to some extent), multiple function approximators can be employed to select a better estimate of the value function rather than having a single estimate. As the function approximates are susceptible to noise, the target Q-value tends to overshoot the true Q-value leading to sub-optimal policy learning. This is still an open research area where the objective is to eliminate the overestimation bias.

Further, to address the problem of poor target function approximation, domain knowledge of the environment can be leveraged to provide more accurate target estimates that closely align with the true value functions [41, 42]. This will assist the agent in accumulating timely rewards, thereby steering the agent's policy toward the optimal one [43]. With respect to disaster response applications, integrating data such as terrain elevation levels, weather conditions, disaster impact region, target characteristics, and human population information can enhance UAV decision-making. This integration can guide the UAVs toward critical regions (the densely populated areas prone to high volume of water accumulation), optimizing the response strategies [44]. However, the use of domain knowledge is not straightforward and heavily depends on the dynamics of the environment. While the incorporation of domain knowledge to enhance RL policies beyond standard algorithms is relatively recent, this research area holds the potential to yield robust RL models [41, 42, 43].

This thesis demonstrates the effective utilization of domain knowledge to guide the learning of RL based policies, especially in the initial phases of training. The proposed approaches seek to expedite reward accumulation while mitigating issues associated with random target function approximation. In the following sub-section, the key challenges involved in learning a multi-UAV Deep RL policy are presented, with a specific emphasis on its application in flood response scenarios.

RL based multi-UAV policies for Flood Disaster Response: Building upon the previous discussions, the key research challenges are outlined as following:

1. Sparse rewards in early training: Obtaining meaningful rewards during the initial stages of training Deep RL models for UAVs in flood scenarios is challenging, often resulting in slow convergence.

2. Initialization of target function approximators: In algorithms such as actor-critic, the random initialization of target functions can result in sub-optimal learning, impeding the swift training of policies.

3. Overestimation bias: Deep RL algorithms can suffer from overestimation bias, where estimated Q-values surpass true Q-values, leading to sub-optimal policies. Mitigating this bias is crucial for accurate learning.

4. Environment modeling: Flood environments exhibit dynamic and stochastic behavior, posing challenges in training Deep RL policies for multi-UAV systems.

5. Lack of generalizability: The learnt policies might struggle to generalize across diverse environments, requiring techniques to ensure learned behaviors apply effectively to new, unseen situations.

6. Handling noisy environments: Real-world environments, including disaster-stricken areas, are often noisy and unpredictable. RL algorithms need to robustly handle noise in sensory inputs for reliable decision-making.

7. Incorporating Domain Knowledge: Leveraging domain knowledge of the environment can be crucial for certain tasks. RL algorithms must effectively incorporate this knowledge to enhance learning and decision-making.

8. Balancing Exploration and Exploitation: UAVs must strike a balance between exploring new strategies (exploration) and exploiting known successful actions (exploitation) to learn suitable policies in limited time-frame due to their finite energy.

9. Minimizing Overlapping Errors for Area Coverage: Precision in coordinated actions is crucial to prevent overlapping errors among UAVs to maximize joint area coverage.

10. Centralized Training Paradigms: Employing a ground control unit (GCU) for aggregating global information enhances the management of multiple UAVs by ensuring coordinated movements and optimized decision-making. However, this dependency on centralized systems pose challenges in scenarios like flood disasters, given that the data is vastly distributed over the environment, complicating its applicability.

11. Decentralized Training Paradigms: Leveraging information sharing among UAVs, decentralized systems facilitate the learning of robust and effective local policies. This approach enhances adaptability and responsiveness, making UAVs well-suited for rapidly changing flood disaster scenarios. However, challenges of slow convergence, lack of coordination and exploration needs to be mitigated to train a decentralized multi-UAV policy.

1.6 Research Focus & Identified Objectives

In this thesis, the research primarily focuses on:

- Understanding the critical role of selecting an RL algorithm for learning autonomous UAV controls depending on the application and the environment.
- Developing Deep RL algorithms for multi-UAV control for disaster response applications, specifically emphasizing on the identification of critical regions in flood-prone areas.
- Incorporating domain knowledge to enhance learning of directed RL policies for flood area coverage and achieving collaborative objectives with autonomous multi-UAV systems.
- Further, the application scope of Deep RL based multi-UAV solutions is extended to address path planning and target tracking problems.

Based on the key research points, specific objectives addressed in this thesis are outlined below.

1. The first objective is to develop Deep RL algorithms enabling UAVs to operate effectively in flood environment, considering both discrete and continuous action

spaces. The task is to identify critical regions in flood-prone areas, utilizing multiple UAVs for optimal coverage.

2. The second objective is to train decentralized multi-UAV policies, enhancing UAVs' deployment efficacy and facilitating flexible and resilient disaster response strategies.

3. The third objective is the identification of serviceable paths to these critical locations for waterborne evacuation vehicle(s) (WBVs) to perform rescue operations during floods.

4. This thesis also addresses the problem of real-time tracking of a moving convoy of vehicles. The final objective is to learn multi-UAV control policies for continuous tracking of a moving convoy.

1.7 Contribution of the Thesis

This thesis aims to demonstrate the effective utilization of domain knowledge to guide the learning of Deep RL based policies, especially in the initial phases of training. The proposed approaches seek to expedite reward accumulation while mitigating issues associated with random target function approximation.
For each objective, the proposed solutions are highlighted as part of the research work, outlined below:

1. Directed Explorations During Flood Disasters Using Multi-UAV System.
2. Continuous Multi-UAV Control with Directed Explorations during Floods.
3. Autonomous Flood Area Coverage using Decentralized Multi-UAV System.
4. Real-Time Serviceable Path Planning during Floods.
5. Autonomous Multi-UAV Control for Moving Convoy Tracking.

In the subsequent sub-sections there is a brief description of each of the proposed solutions.

1.7.1 Directed Explorations During Flood Disasters using Multi-UAV System

This contribution proposes a Deep RL based method to learn a control policy for a multi-UAV system in order to perform non-overlapping coverage of critical regions in a flooded area. To guide the initial exploration, water flow estimates extracted from the D8 algorithm [45] are used to train the UAV policy. The proposed solution helps in managing the actions of the UAVs in accordance with the water flow dynamics to expedite the training process. The objective of the UAVs is to find maximum number of critical regions, i.e., densely populated areas that are prone to high volume of water accumulation. Since a UAV can only observe partial information from the environment at any given point in time, the D8 flow algorithm maps the flow direction by taking into consideration the terrain elevations of the flooded area. The reward function is designed in such a manner so as to avoid collisions and repeated observations by the UAVs. This encourages the UAVs to visit unobserved locations which in turn increases the area coverage and potentially identifying a larger number of critical regions. The proposed algorithms, named, D8-Q-learning (D8QL) and D8-Deep-Q-network (D8DQN) are assessed in diverse environments. The performance of the proposed algorithms is compared with various other UAV-based area coverage approaches from the literature. Further, to test the generalizability of the proposed algorithms, the learnt multi-UAV policies are executed in unseen flooded environments and their performances are evaluated.

1.7.2 Continuous Multi-UAV Control with Directed Explorations during Floods

In this contribution, an extension to the previous work [44] is presented, where the limitation of DQN based model is addressed which limits the UAVs to a finite and discrete set of actions. Consequently, maneuvering UAVs smoothly becomes challenging due to the limited action space. Further, as the employed exploration algorithm (D8) in [44] works based on the surface elevation information, the UAVs are prone to clustering at nearby sub-regions having low elevation. Such clustering restricts the UAVs' coverage to a relatively smaller region rather than having a broader coverage of the environment, which could have helped the UAVs in identifying a larger number of critical regions during floods. To address these limitations a Deep RL algorithm, named, D3S is proposed with the objective to learn continuous actions for the UAVs with non overlapping trajectories. The task remains the same, that is identification of critical regions in a flooded environment. To learn a continuous control policy for the UAVs, Deep Deterministic Policy Gradient (DDPG) [32] algorithm is employed with an improved target Actor. The proposed target actor guides the UAV exploration using the D-infinity (DINF) [46] water flow algorithm. DINF maps the water flow direction from a given location to its neighbouring areas based on the triangular facet theory. Using this information, the UAVs' actions are tweaked in the direction of the flow estimate. Prior to the empirical evaluation of the proposed algorithm, it is hypothesized that these flow estimates contribute to learn a more efficient multi-UAV policy by boosting the target actor of DDPG. Additionally, the implementation of Path scatter (a dispersion strategy for UAVs) restricts UAV actions, preventing clustered formations, and aids the multi-UAV system in maintaining inter-UAV separation for enhanced coverage.

1.7.3 Autonomous Flood Area Coverage using Decentralized Multi-UAV System

As discussed, majority of the prominent work done in the field of multi-agent reinforcement learning (MARL) considers some form of centralized entity to make individual agents learn from global knowledge of the environment [47, 48, 49]. The environment related

information is gathered from each agent and stored in a centralized unit. However, utilizing a centralized system to train a multi-UAV policy during natural disasters could be limiting due to the widespread distribution of data throughout the environment. Further, the location of the centralized entity should be known by the UAVs at all times as they need to have bi-directional communication with the central server irrespective of their distances. This becomes very restrictive for the UAVs to operate when the environment is large, dynamic and stochastic [18].
Considering this problem, a more flexible approach is proposed to learn a multi-UAV policy using decentralized training paradigm. In literature, various researchers have attempted to solve dec-POMDP problems using decentralized Deep RL approaches [50, 51, 52, 53]. Motivated from the prevalent literature work, a decentralized Deep RL algorithm is proposed to train a multi-UAV system to operate effectively in a highly dynamic flood environment. The objective is to gather critical ground information of a flooded area using multiple autonomous UAVs with limited energy for relief and evacuation purposes. Further, opportunistic communication among UAVs is enabled so as to exchange their experiences in order to improve the local UAV policies. In addition, Coverage Maps are introduced to gain insights into UAVs' observation histories, encouraging them to have non-overlapping trajectories for maximizing overall coverage.

1.7.4 Real-Time Serviceable Path Planning during Floods

After identifying critical regions in the flood-struck area, the problem of path planning is addressed to assist the waterborne evacuation vehicles (WBVs) to reach these locations to evacuate victims. A team of autonomous UAVs is deployed to perform cooperative sensing and coverage of a flood-struck region to identify serviceable paths to reach these critical regions from the location of the WBV. A path is said to be serviceable when it is clear of obstacles and shallow water, for possible movement of WBVs. However, autonomous navigation and formation control of multi-UAV systems pose a significant challenge for the robotic systems that operate in partially-observable, dynamic and continuous environments.
To address this, a Deep RL algorithm is employed to learn a cooperative multi-UAV policy for coverage and formation control. The coverage information provided by the UAVs capture the presence of obstacles present over the shortest path connecting the start and target location. To provide connected coverage, the UAVs maintain an end-to-end formation to prevent any coverage gaps. This coverage information is utilized by the proposed path planning algorithm, named, MEA*, to minimize the number of expansion nodes and realize a serviceable path quickly.

1.7.5 Autonomous Multi-UAV Control for Moving Convoy Tracking

This thesis also addresses the problem of tracking a moving convoy of vehicles using multiple autonomous UAVs, where continuous tracking is ensured by maintaining the convoy within the joint Field-of-View (FoV) at all times. To learn an autonomous policy for the multi-UAV system, a Deep RL algorithm, named, GPR-MADDPG is proposed which employs Gaussian Process Regression (GPR) to approximate the target Q-value function. Further, the GPR model's kernel function is adapted to address the excessive variance in the trajectory estimation. The reward function is designed to maximize convoy coverage by optimizing FoV overlaps while minimizing tracking errors. Experiments were performed on road trajectories of varying complexities generated using OSRM tool [54], along with varying convoy speeds and the number of UAVs. Further tests were performed using a 3D physics simulator, known as Gazebo [55]. The experiments show that the proposed GPR-MADDPG model results in the least amount of overlapping error and accumulates maximum rewards as compared to other prevalent approaches in the literature.

1.8 Outline of the Thesis

The above discussed contributions are presented as individual chapters in this thesis. An outline of all these chapters is provided below:

- **CHAPTER 1:** It presents an introduction to UAVs, background on reinforcement learning and provides a brief overview of its algorithms. It explores the motivation behind leveraging domain knowledge and enhancing function approximators to advance Deep RL for multi-UAV controls, particularly for the application of flood area coverage and object tracking.
- **CHAPTER 2:** In this chapter a comprehensive overview of the related literature is provided pertaining to autonomous UAV control across diverse applications. Additionally, the challenges and limitations in the related literature are discussed to highlight the research gaps.
- **CHAPTER 3:** In this chapter, a Deep RL algorithm, named D8DQN is proposed that leverages domain knowledge using D8 flow estimation algorithm. This method guides Deep-Q Network (DQN) exploration, enabling autonomous multi-UAV control for flood area coverage and identification of critical regions.
- **CHAPTER 4:** Extending the work proposed in Chapter 3, continuous multi-UAV controls are learned using the proposed D3S algorithm. D3S uses D-infinity algorithm to enable directed explorations based on the exact degrees of water flow estimates. Additionally, Path scatter is introduced for broader area coverage.
- **CHAPTER 5:** To achieve multi-UAV autonomy in environments with distribution information, a fully decentralized training paradigm is introduced to learn autonomous multi-UAV controls. To expedite learning, inter-UAV communication is enabled and Coverage Maps are used to prevent frequent visits of UAVs to similar regions.
- **CHAPTER 6:** In this chapter, the problem of path planning is addressed to identify serviceable paths for waterborne vehicles during floods. Autonomous UAVs

collaborate to provide real-time data on obstacles and shallow water regions. This information is utilized by the path planning algorithm (MEA*) to minimize the number of expansion nodes in A* and expedite the identification of serviceable paths.

- **CHAPTER 7:** In this chapter, the focus is on developing autonomous control policies for UAVs to track and maintain a moving convoy within their joint Field-of-View (FoV). The continuous coverage is achieved by maintaining the convoy within the joint FoV at all times. This is achieved by introducing a Gaussian Process Regression (GPR) based value function approximator to train Deep RL multi-UAV policies for tracking a moving convoy. GPR provides with a continuous estimate of the target critic Q-value while adapting the kernel function to manage the high variance.
- **CHAPTER 8:** This chapter marks the conclusion of the thesis, summarizing the contributions made throughout the research. Additionally, it provides insights into the potential future directions for this work.

2| Literature Review

This chapter focuses on providing a comprehensive review of various control methodologies to learn autonomous policies for UAVs. It delves into model-based and model-free RL approaches, alongside non-RL methodologies such as rule-based systems, heuristic approaches, meta-heuristic algorithms, and non-RL model-based methods. This chapter aims to offer a thorough overview of these prominent algorithms, outlining their strengths, weaknesses, and applications across various domains requiring autonomous multi-UAV solutions. Additionally, the chapter discusses the challenges and limitations of these algorithms to highlight research gaps. By comprehending the latest advancements in control algorithms, researchers, practitioners, and policymakers can develop and apply efficient control strategies to achieve UAV autonomy for disaster response applications.

2.1 Overview of Reinforcement Learning Algorithms

This section provides an in-depth discussion on fundamental and prominent RL algorithms in both discrete and continuous action spaces, offering a comprehensive overview of popular algorithms. This discussion forms the basis for understanding the potential applications of these algorithms in multi-UAV systems. RL algorithms can be primarily categorized into two groups: model-based and model-free algorithms [19].

2.1.1 Model-based Algorithms

The model-based algorithms infer the model of the environment from its observations and then plan a solution using that model. The two most popular model-based RL algorithms are policy iteration and value iteration. Both these algorithms can be directly applied to the MDP quintuple $< S, A, T, R, s_0 >$ to learn a control policy. In policy iteration, the policy π is initialized arbitrarily, and the value function $V(s)$ is set to either zero or assigned random values. The first step is to perform policy evaluation for the value function V(s) under the current policy π [56]. This can be done using the Bellman equation, given as:

$$V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')] \tag{2.1}$$

where γ is the discount factor. $\pi(.)$ outputs an action a (that could be UAV control values for throttle, yaw, pitch and roll actions) in state s (usually given by the current location of the UAV in the environment) and $P(., .|., .)$ is the probability of transitioning to state s' (i.e., moving to a different location) when taking action a in state s. r is the reward received by the UAV for taking action a in state s and transitioning to state s'. The reward function is usually based on the objective of the problem that is being considered [57]. Next, the Policy Improvement step is performed to update the policy π by selecting the action that maximizes the expected value of taking that action in the current state, given as:

$$\pi(s) \leftarrow argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] \tag{2.2}$$

both these steps are repeated until the policy converges [58]. Another model-based algorithm known as value iteration can be also applied to learn autonomous control policies for agents [19]. The value iteration algorithm can be applied iteratively to realize the optimal actions corresponding to each state of the environment. The value iteration update equation is given by:

$$V(s) \leftarrow max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] \tag{2.3}$$

where $V(s)$ is the value function, a denotes the action, s' is the next state, $p(s', r|s, a)$ is the transition probability, r is received reward, and γ is the discount factor. Once the value function has converged, the optimal policy is given by:

$$\pi(s) = argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] \tag{2.4}$$

In the above discussed algorithms there is a significant limitation i.e., the dynamics/model of the environment should be known *a priori* for their applicability. However, in a majority of applications, like utilizing UAVs for area coverage during floods and employing them in real-time path planning applications to assist evacuation teams in reaching victims, the dynamics of the environment are usually unknown. For such intricate tasks, model-free RL approaches are used, learning control policies directly for the experiences obtained by interacting with the environment, without relying on the model of the environment.

2.1.2 Model-free Algorithms

In contrast to model-based, model-free algorithms focus on learning the consequences of actions through experience [19]. These algorithms repeatedly perform actions, adjusting their policy to maximize rewards based on observed outcomes. In the subsequent discussions prominent model-free RL algorithms are explored.

Q-Learning: This is a model-free, off-policy RL algorithm that learns the optimal actions corresponding to each state of the agent that it can attain in the environment. Off-policy methods are the ones where the evaluated or improved policy is different from the one that is used to generate the data (i.e., the behavioural policy). In contrast, on-policy methods attempt to evaluate or improve the policy that is used to make decisions. Q-Learning is a tabular algorithm [59] where the Q-values are computed corresponding to every action in each state. Policy training is performed using an $\epsilon - greedy$ strategy, given as:

$$a_t = \begin{cases} \underset{a \in A}{max}\ Q_t(s_t, a_t) & with\ probability\ 1-\epsilon \\ random\ action & with\ probability\ \epsilon \end{cases} \tag{2.5}$$

where, $0 \leq \epsilon \leq 1$ and A denotes the feasible action set. The $\epsilon - greedy$ strategy is known as the exploration-exploitation strategy where in the initial phase of training the agent randomly explores the environment to gain experience. After enough experience is obtained, the agent selects the greedy action in every state based on learned Q-values to maximize its rewards in the long run. Further, $\epsilon - decay$ strategy is adopted where at the beginning of training a high value of ϵ is selected (i.e., close to one) to prompt random exploration and as the estimation of the Q-value function improves with time, the ϵ decays, increasing the probability of selecting an action based on max Q-value rather than a random action.
Q-learning update equation is given as [19]:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma \underset{a'}{max}\ Q(s', a') - Q(s_t, a_t)] \tag{2.6}$$

where α is the learning rate and a' denotes the action on seeing state s' under the current policy. However, in large state-space environments, Q-learning becomes impractical as it becomes infeasible to generate a Q-value table containing values for every state-action pair. To address this limitation a neural network based Q-value function approximation technique was proposed known as Deep Q-networks.

Deep Q-networks (DQN): This is [13] is also a model-free off-policy learning algorithm that uses a pair of neural networks known as a learning network and the target network. Both these networks are initialized with the same weights but are updated at different frequencies. Each network takes the state of the agent as input and approximates the Q-value function corresponding to each feasible action that the agent can take. DQN is also based on $\epsilon - greedy$ strategy where initially the agent explores the environment based on random actions. As the agent accumulates adequate experience, this gathered information ($< s_i, a_i, r_i, s_{i+1} >$ stored in a buffer) is used to train the learning network. Random samples in form of mini-batches are extracted from the experience replay buffer for training. During training, the mean square loss is calculated between the Q-value of the learning network and target network and then gradient descent step is performed w.r.t. weights of the learning network [13], given as:

$$\nabla_{\theta_t} \mathcal{L}_t(\theta_t) = \mathbb{E}_{s,a \sim \rho(.); s' \sim p(s'|s,a)} \left[\left(r + \gamma \underset{a'}{max}\ Q(s', a'; \theta_{t-1}) - Q(s, a; \theta_t) \right) \nabla_{\theta_t} Q(s, a; \theta_t) \right] \tag{2.7}$$

where t denotes the iteration and $\rho(.)$ is a probability distribution over sequences s and actions a. However DQN works well only with discrete and finite action spaces since it calculates the maximum Q-value by assessing all feasible actions, limiting its use in continuous action spaces.

Deep Deterministic Policy Gradient (DDPG): Similar to Q-learning and DQN, DDPG [32] is also a model-free off-policy learning algorithm. DDPG is based on an Actor-Critic framework where the actor-network generates real-numbered action values for the agent based on its current state. The critic network takes this current state and the action generated by the actor as input and provides a Q-value estimate defining a quantitative measure of the suitability of the action in a given state. Target actor and critic networks are also part of the DDPG architecture to stabilize policy learning. The network weights are randomly initialized and before training, the agent performs random actions to gather experience from the environment. However, unlike random exploration in DQN where an action is selected randomly among the feasible actions using the $\epsilon - greedy$ strategy, DDPG uses additive noise that is added to the generated actions to achieve random exploration. The Q-value of the target critic network is calculated as [32]:

$$y_k = r_k + \gamma Q'(s_{k+1}, \mu'(s_{k+1}|\theta^{\mu^{'}})|\theta^{Q^{'}}) \tag{2.8}$$

where $Q'(.,.|.)$ denotes the target critic network and $\mu^{'}$ denotes the target actor network. $\theta^{\mu^{'}}$ denotes the target actor network weights and $\theta^{Q^{'}}$ represents the target critic network weights. The weights of the learning actor network are updated using the sampled policy gradient, given as:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_{k} \nabla_a Q(s, a|\theta^Q)|_{s=s_k, a=\mu(s_k)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_k} \tag{2.9}$$

where k denotes the time-step of the transition as sampled from the experience replay buffer $< s_k, a_k, r_k, s_{k+1} >$ and N denote the size of the mini-batch sampled for training.

All the above discussed algorithms are limited to the autonomous functioning of a single agent (UAV), however, to address large-space environment challenges multi-agent systems are employed. Below, a notable multi-agent reinforcement learning algorithm suited for continuous state and action spaces is discussed.

Multi-agent Deep Deterministic Policy Gradient (MADDPG): This algorithm [60] works in a manner where each agent (UAV) is controlled by a separate actor network that learns the policy for each individual UAV and uses the sampled policy gradient method for updating the network parameters. MADDPG trains a centralized critic (another deep neural network) that learns the $Q(s, a)$ for each agent. Target networks are used to stabilize the training. In a system of n UAVs, each UAV i selects an action a_i w.r.t. to the current policy and similarly, n actions are executed $a = (a_1, a_2, ..., a_n)$. The UAVs' experiences are stored in a single replay buffer $(S, S', a_1, a_2, ..., a_n, r_1, r_2, ..., r_n)$, where S denotes the initial state (i.e., local observations of each UAV) and S' denotes the next state. In each epoch of training, a random mini-batch (m) is selected from the replay buffer. The critic network is updated based on the following loss equation:

[60]:

$$\mathcal{L}(\theta_i) = \frac{1}{m}\sum_t \left(y^t - Q^i_\mu(S^t, a^t_1, a^t_2, ..., a^t_n)\right)^2 \tag{2.10}$$

where θ_i are parameters of UAVs' policies μ_{θ_i} and Q^i_μ is the centralized Q function.

$$y^t = G^t + \gamma Q\mu'(S'^{\,t}, a'_1, a'_2, ..., a'_n)|_{a'_k=\mu'_k(o^t_k)} \tag{2.11}$$

The actor-network parameters are updated using sampled policy gradient given as [60]:

$$\nabla_{\theta_i} J \approx \frac{1}{m}\sum_t \nabla_{\theta_i}\mu_i(o^t_i)\nabla_{a_i}Q^\mu_i(S^t, a^t_1, a^t_2, ..., a^t_n)|_{a_i=\mu_i(o^t_i)} \tag{2.12}$$

However, learning a suitable policy is not a trivial task as there are many system and environment-related constraints and control parameters that needs to be considered. Many recent studies [41, 42, 43] have used domain knowledge for policy learning. This form of learning is specific to the application where it's being applied and has shown improved performance with better policies. With the utilization of domain knowledge, the UAVs can learn from the available information about the environment to improve their current policy in a brisk fashion and perform the task at hand [41]. However, exploiting domain knowledge and utilizing it to learn an optimal control policy for multi-UAV systems is still a rather new and ongoing research area [41, 61].

2.2 RL-driven UAV Applications

In this section, the related literature w.r.t. the application of RL and Deep RL algorithms in the context of multi-UAV systems is discussed, particularly in response applications such as area coverage, path planning, and target tracking.

2.2.1 Area Coverage during Disasters

Considering an MDP quintuple $< S, A, T, R, s_0 >$ for area coverage problem, where $S : \{s_1, s_2, ..., s_n\}$ denotes the state containing the observations of n UAVs, where s_i denotes the information corresponding to the location and the battery level of i^{th} UAV. Additional information corresponding to water levels and the number of victims in the regions sensed by the UAVs can be incorporated as part of the state to prompt the UAVs to focus more on such critical regions. $A : \{a_1, a_2, .., a_n\}$ denotes the actions of the UAVs, where a_i lies within a feasible action range. $P_{ss'}$ is the transition model of the environment. R is the reward function denoting the incentives received by the UAVs that are used as feedback to learn an optimal policy. s_0 is the starting configuration of the UAVs in the environment. To address the task of area coverage using multiple UAVs the objective function is usually formulated around maximizing the coverage area or maximizing the coverage of target regions in a limited time frame due to the finite battery life of UAVs. So, the reward function can be formulated around the number of cells sensed by the multi-UAV team in an episode or in scenarios where specific regions are more important than other regions, the reward function can be weighted average over the number of cells and type of cells the UAVs cover.

Algorithms such as Q-leaning [62] (for relatively smaller area coverage tasks) and DQN [44] (for large or continuous state-space environments) can be employed to address area coverage problems using UAV, where UAV actions are limited to a discrete and finite set. Authors in [29] employ a double deep Q-network (DDQN) model to learn UAV controls, where the UAV is tasked to maximize coverage under varying power constraints. Further, multiple options for landing positions in the simulated environment are considered along with no-fly zones. To train the DDQN model, map-based spatial information is fed as input to the convolutional network layers of the model. Authors in [63], address the problem of covering vast areas using UAVs by learning autonomous controls with the help of an actor-critic algorithm. Further, to reduce UAV energy consumption, such actions are selected that greatly reduce energy loss. Pham et al. [64] proposed a function approximation-based RL method for locating the missing humans in a post-disaster scenario using UAV along with flight control. Detailed implementation of the UAV performing a search-and-rescue task is highlighted where the UAV is able to locate the human's location from an arbitrary starting point. However, their work relies on a tabular RL method to learn the Q-value functions. Such an approach would fail to generalize in environments with continuous state and action spaces.

In case of multi-UAV systems, multiple Q-learning or DQN models (individual to each UAV) can be employed with a centralized training unit. In [28], authors employed a Deep RL technique known as Deep Q-Networks (DQN) for trajectory planning of multiple UAVs for flood monitoring tasks. It was assumed that UAVs have infinite battery life and a UAV is able to gather information related to the heading angle and bank angle of other UAVs, however, no communication protocol was employed. For continuous action space problems (where UAVs' action lies over a continuous range) DDPG algorithm [35] can be used to learn autonomous policy in case of a single UAV and to train multiple UAVs jointly, MADDPG and MATD3 algorithms [65] can be employed.

Additionally, in the case of multiple UAVs working in the same environment, it's important to minimize the sensing overlap among the UAVs to maximize the coverage. This can also be included as a key component in the reward function [44]. However, when deploying multi-UAV systems, several challenges needs to be addressed. Controlling all the UAVs simultaneously through a centralized server/ground control unit (GCU) proves to be a challenging task [52]. Especially when the information is distributed throughout the environment, it becomes difficult to maintain bi-directional communication between the GCU and the UAVs over large state-space environments. Further, as the UAVs have a finite flight time due to limited battery life, it becomes opposing for the UAVs to explore the environment and stay connected with the GCU at all times due to the limited communication range. In centrally controlled systems, it's important that all the UAVs send back their current state to the GCU and based on the current RL policy the action commands are sent back to each UAV [66].

To overcome this problem, decentralized multi-UAV systems are employed that are capable of learning control policies without the intervention of GCU [18]. In such a setting, the UAVs communicate with others to exchange coverage and trajectory information to maximize the objective of area coverage [67]. By adopting this framework the UAVs are not dependent on the GCU to learn its control policies and trains directly from its local experience and the experience gained from communicating with other UAVs. However, as the quality of control policy learned by a UAV is mostly dependent on its local experience (as communication can only work over a limited range and the energy consumed during communication will reduce the UAVs flight time), it usually takes much longer to train a decent policy in a decentralized setting as compared to a centralized one. Another problem in disaster scenarios is the continuously changing dynamics of the environment, such as in the case of active floods. To address this problem, it's important to generalize the state space and incorporate the environment parameters to train a robust policy.

2.2.2 Path Planning

Path planning is a well-known and extensively studied application where an agent (UAV) needs to navigate through the environment to reach the target location(s). The most prominent path-planning algorithms are A* and RRT and their variants that are extensively applied to realize cost-effective paths for UAVs through obstacles and obstacle-free environments [68, 69]. However, in the context of dynamic environments that are continuously changing, it's not feasible to pre-determine the trajectory of the UAVs. To address this problem, UAV-based real-time sensing of the environment is adopted in many path-planning applications [70, 71]. However, relying on pilot-controlled UAVs may be ineffective, as it is not always possible to keep the flying UAV in the line of sight [72]. Hence, autonomous UAVs are employed to cooperatively survey/monitor the region from a start location to the target location to warn against any foreseen obstacles or unserviceable regions in case of disasters, such as floods [73]. Considering the same MDP quintuple as defined in subsection 2.2.1, the reward function can be tweaked to provide incentives corresponding to covering connected regions between start and target locations to identify a serviceable path. Further, the additional incentive can correspond to UAVs maintaining an interleaved formation without collision to prevent coverage gaps.

However, the use of autonomous UAVs brings along challenges such as energy management, cooperative sensing and training, non-overlapped coverage and most importantly, achieving the objective in a short time-frame due to energy limitations. In [74], authors introduce an enhanced Q-Learning algorithm employing Artificial Neural Networks (ANNs) for the optimization of UAV swarm path planning. They encode the action space as a discrete list of values, encompassing all feasible movements. The reward function is designed to promote increased exploration of unobserved regions while discouraging frequent revisits to similar areas. In [70], authors propose a Deep RL based approach for UAV path planning using global observation information. A set of observed global maps are provided as input to the dueling double deep Q-network (D3QN) algorithm to approximate the value function corresponding to all feasible actions. The $\epsilon - greedy$ strategy of D3QN is integrated with heuristic search rules to select the appropriate action. In [75], author investigated vision-based UAV navigation in GPS-denied virtual environments. A Variational Autoencoder (VAE) is employed to enhance sample efficiency and a Proximal Policy Optimization (PPO) agent is used that is capable of tracing rivers in realistic photo simulations. The reward function is structured to incentivize the UAVs to maintain proximity to the center of the river while flying. The proposed approach is compared with another agent trained with Imitation learning (IL). Authors in [76], addresses the problem of guiding a UAV along desired paths using modified DDPG algorithm. A double experience replay buffer (DERB) is used to expedite the training process. The reward function is deigned to minimize the cross-track error. Extensive experiments are carried out to emphasize the effectiveness of the proposed DERB-DDPG approach.

2.2.3 Object Tracking and Formation Control

Active object tracking using a UAV concerns with maintaining the target object within the field of view (FoV) of the UAV at all times [47]. Analysing the change in the orientation and speed of the object and controlling the UAV actions accordingly is generally the process followed to achieve this objective [77]. However, not knowing the trajectory of the moving object *a priori* leads to failure as the object moves out of the FoV of the UAV and it becomes difficult for the UAV to relocate the object. Resolving the problem of relocating the object is usually achieved by adjusting the altitude of the UAV to enlarge the FoV. However, this approach leads to image-resolution problems, along with high energy consumption, and if multiple UAVs are involved, it could lead to the dissolution of their cooperative formation. Another challenge lies in the complexity of learning the trajectory of a moving object. There are few studies in the literature focused on estimating the UAV trajectory by utilizing the observed path of a moving object. However, depending on the change in the speed of the object and sharp changes in orientation (maybe due to curves in the path), the estimation can be very challenging. The reward function for an RL algorithm for such a problem can be designed in a manner where the incentive is proportional to the distance between the center of the FoV of the UAV and the position of the target, so as to maximize the amount of time during which the object is in the sight of the UAV [78]. An extended application of this is to track multiple objects moving in some formation using multiple UAVs where the objective is to maintain all the objects in the joint FoV at all times [47, 78]. This brings in the additional challenge of how to manage multiple UAVs simultaneously and also prevent them from colliding.

In [47], authors propose a MADDPG based model where the UAVs are tasked with tracking of moving objects while maintaining an optimal formation. A well-designed reward function is formulated to incorporate tracking, formation control and collision avoidance objectives in policy learning. In [77] authors propose a multi-agent deep reinforcement learning algorithm to learn cooperative tracking policies for UAVs using a reciprocal reward strategy to track moving targets. The proposed approach reshapes the UAV reward to a regularized value depending on the rewards of the neighbouring UAVs. A UAV is assumed to have bi-directional communication with its neighbouring UAVs. Further, UAVs are assumed to have infinite energy for maneuvering and performing communication. Li et al. [79] addresses the problem of deficient learning of a Deep RL model due to the lack of knowledge for initial parameter setting and the incapability of the model to generalize in different environments. To tackle this, a Deep RL based Meta Twin Delayed DDPG (Meta-TD3) model is proposed to obtain parameters of the neural network and use it as prior knowledge for UAV maneuvering and target tracking. By employing TD3, the authors were able to eliminate the overestimation bias in DDPG to some degree, but TD3 further brings underestimation bias into the picture which is not addressed in [79]. Authors do aim at learning a UAV policy that is well suited to multiple tasks but fails to examine the scalability of their algorithm and also the robustness of the model, as no physical parameters of real-world environment are considered. Authors in [80] present a UAV based model for persistent target tracking in the presence of obstacles using a Deep RL based technique called Target Following Deep Q-Network (TF-DQN) implemented in an urban setting. The proposed work [80] lacks scalability when there are multiple targets to track using multiple UAVs. Further, the actions of the UAV are limited to a discrete set which limits the applicability of such models when it comes to deployment.

2.3 Non-RL Approaches

This section discusses the non-RL approaches that are proposed and adopted in the literature to optimize autonomous control policies for UAVs.

Rule-Based Systems: Rule-based systems define UAV actions through predefined rules based on their observations [81, 82, 83]. Behavior trees are utilized for hierarchical organization of rules and have proven to be effective in applications such as UAV coverage strategies and area mapping [84, 85]. Further, Expert systems, resembling human decision-making are adopted in the literature for UAV patrolling and emergency response tasks [86, 87]. Furthermore, Fuzzy logic systems are utilized for managing uncertainty present in the data captured by UAVs[88, 89].

Meta-heuristic Approaches: These approaches, especially the evolutionary algorithms like Genetic Algorithm (GA) [90, 91], Differential Evolution (DE) [92], Particle Swarm Optimization (PSO) [6, 93], and Ant Colony Optimization (ACO) [94, 95] are extensively employed to optimize UAV control parameters and learn optimal actions. These techniques address trajectory optimization problems and can handle constraints such as UAV(s) altitude, bank-angle, and energy. They have also been employed for tasks like target localization [96], collision-free path planning [97], and coordinated disaster inspection using multi-UAV systems [6].

Model-Based Approaches: Model-based techniques like PID controllers [47, 98, 99], regulate UAV movement by utilizing parameters such as location, velocity, and heading angle, along with sensor feedback. Relying on sensor data aids the PID controller in comprehending the environmental dynamics, and to make real-time corrections. The Model Predictive Control (MPC) approach [100, 101, 102] uses mathematical models to predict future behavior and optimize control signals over a time horizon. Another approach, named, Model Reference Adaptive Control (MRAC) [103, 104] is employed to adjust UAV action commands based on its current state and aligning w.r.t. a reference model. These methods are applied for tasks such as trajectory planning, target tracking, and swarm-based exploration in GPS-denied environments.

2.4 Chapter Summary

In this chapter, the existing literature on autonomous Unmanned Aerial Vehicles (UAVs) is explored, with a specific focus on disaster response applications. The primary emphasis is on identifying techniques used for learning autonomous control policies for UAVs, particularly Reinforcement Learning (RL) algorithms, due to their applicability in stochastic and dynamic environments. The discussions are centered around model-based and model-free RL algorithms, along with non-RL approaches. This chapter aims to help researchers, practitioners, and policymakers in implementing effective strategies for autonomous multi-UAV control in disaster response applications. Additionally, it identifies research gaps, guiding further exploration and development in UAV autonomy for disaster management.

3| Directed Explorations During Flood Disasters Using Multi-UAV System

To effectively execute disaster relief operations during floods, quick and accurate information about the flooded areas is crucial. However, to perform data-gathering tasks using autonomous UAVs in unknown environments, it is important to ensure that correct autonomy is in place. Reinforcement learning (RL) algorithms, specifically Deep RL methods, provide suitable control policies by learning complex value functions over large state and action spaces.

In this chapter, a domain knowledge-based exploration strategy is proposed to expedite the training process of a Deep RL algorithm and to accumulate high rewards especially in initial episodes. By adopting this approach, random exploration is restricted, guiding the UAVs toward critical regions during floods. Domain-specific knowledge such as population density, water flow dynamics, and terrain's digital elevation model is encoded as part of the observed state to improve the UAV policy. The proposed approach enables the multi-UAV system to perform non-overlapping coverage of critical regions, integrating domain knowledge of water flow estimates of the flooded area to direct UAV explorations. Further, the reward function is designed to encourage UAVs to visit unobserved locations by preventing repeated observations. The proposed solutions, D8-Q-learning and D8-Deep-Q-network, are evaluated across multiple environments and compared with prevalent approaches in the existing literature. These comparisons highlights the models robustness and improved performance across various metrics.

The rest of the chapter is organized as follows. In section 3.2, the environment description is provided, followed by section 3.1 presenting the system model. section 3.3 introduces the proposed D8QL and D8DQN algorithms for multi-UAV based flooded area coverage. section 3.4 covers the discussion on experiments and results. Finally, in section 3.5, a concise summary of the chapter is provided.

3.1 Environment Description

The environment is a 2D terrain as seen in Figure 3.1(a). It is visualized as a grid with $(m_1 \times m_2)$ cells, where the number of cells in the grid depends on the altitude of the UAVs. The size of each cell is equal to field of view (FoV) of the UAV as shown in Figure 3.2. A 2D grid mask representing water is defined over the environment grid to simulate floods (see Figure 3.1). There are two parameters that define the surface water, one being the water level h and the other being the water flow rate f_{rate}. These parameters help in properly defining the flood water that is in motion. The water level is sensed by preforming bathymetry [105]. Further, a Critical Level $\mathbb{Z}_c$ is defined for each location/cell

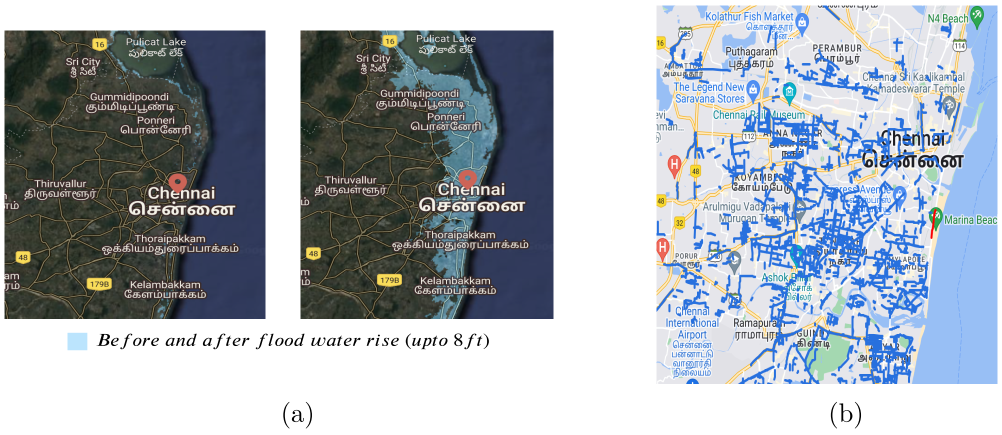


Figure 3.1: (a) 2D Map imagery of Chennai region before and after induced floods (b) Inundation map of the Chennai region (validated with OSM [1]).

c in the environment, given as:

$$\mathbb{Z}_c = h_c \times \mathbb{P}_c \tag{3.1}$$

where $\mathbb{P}_c$ represents the population density level of that cell. Both h_c and $\mathbb{P}_c$ are defined over a discrete and finite range.

3.2 System Model

In the system model, there exists a group of n UAVs(U) : $\{u_i | i \in \{1, 2, ..., n\}\}$ that are tasked to perform optimal area coverage of the flood region.
The area coverage denotes the number of unique grid cells that are covered by the multi-UAV system until their battery runs-out. The UAVs are positively rewarded based on the amount of new information they collect. Hence, the objective of the multi-UAV system is to maximize the overall information gain under the constraint of limited energy of the UAVs. The energy depletion of the UAV is based on its flight time.

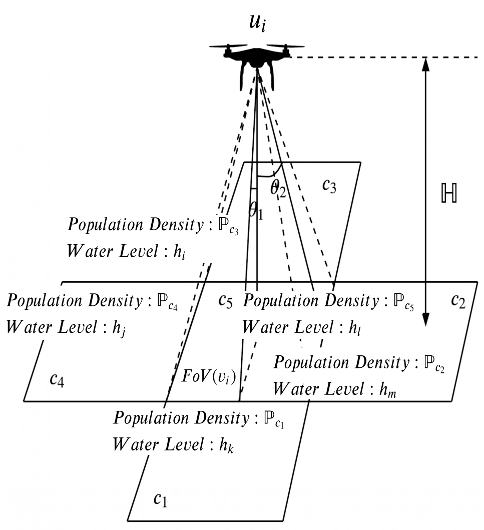

u_i
$\mathbb{H}$
θ_2
θ_1
c_3
Population Density : $\mathbb{P}_{c_3}$
Water Level : h_i
Population Density : $\mathbb{P}_{c_4}$
Water Level : h_j
c_5 Population Density : $\mathbb{P}_{c_5}$
Water Level : h_l
c_2
c_4
$FoV(v_i)$
Population Density : $\mathbb{P}_{c_2}$
Water Level : h_m
Population Density : $\mathbb{P}_{c_1}$
Water Level : h_k
c_1

Figure 3.2: Field of view of i^{th} UAV on the 2D grid.

The energy $\psi_t^{u_i}$ of a UAV at any time t, is calculated as:

$$\psi_{t+1}^{u_i} = \psi_t^{u_i} - \Delta\psi \tag{3.2}$$

where $\Delta\psi$ represents the energy depletion per unit of time.

The underlying idea is to encourage the UAVs to capture unobserved cells rather than revisiting the same cells observed earlier. Hence, the total information received by the UAV from a location/cell is a combination of the $\mathbb{Z}_c$ of that cell and the penalty imposed due to any repeated observations. The penalty is calculated based on the time elapsed since the last observation of that particular cell. The information gain from a cell c by a UAV u_i is given as:

$$I_{c,t_j}^{u_i} = \frac{\mathbb{Z}_c}{max(\mathbb{Z})} \cdot \frac{t_j - t_c}{t_j} \tag{3.3}$$

where, $I_{c,t_j}^{u_i}$ is the information gain from cell c at the current time-step t_j. t_c denotes the last time-step at which the cell c was observed. $\mathbb{Z}_c$ is the critical level of location c.

Next, the proposed solution is described to learn the multi-UAV policy for maximizing the information gain using the D8 flow algorithm.

3.3 Proposed Methodology

This section begins with the presentation of the MDP, followed by the D8 flow estimation method. Subsequently, the proposed solution is discussed to learn a multi-UAV policy through directed explorations.

3.3.1 MDP Formulation

An RL model can be described using a Markov Decision Process (MDP) quintuple : $< S, A, P_{ss'}, R, s_0 >$, where S represents the set of states of the system, A represents the set of allowed actions of the UAVs in any state $s \in S$, $P_{ss'}$ represents the model of the environment with transition from one state to another, R is the reward function and s_0 is the starting configuration of the UAVs.

A state $s \in S$ of a UAV u_i is represented as:

$$s^{u_i} : \{\mathbb{P}_{x+k_1,y+k_2}^{u_i}, e_{x+k_1,y+k_2}^{u_i} | \ \forall \ k_1, k_2 \in \{-1, 0, 1\}\}$$

where $\mathbb{P}_{x,y}$ corresponds to the population density of the grid cell $c : \{x, y\}$ and $e_{x,y}$ corresponds to the terrain elevation of that cell captured from the elevation maps (refer Figure 3.4(a)). The set $\{-1, 0, 1\}$ represents the elevations and population densities of the 8 neighbouring cells to cell c (the cells at the border have only 3 neighbouring cells).

A feasible action $a^{u_i} \in A$ of a UAV u_i is either the movement to one of the neighbouring

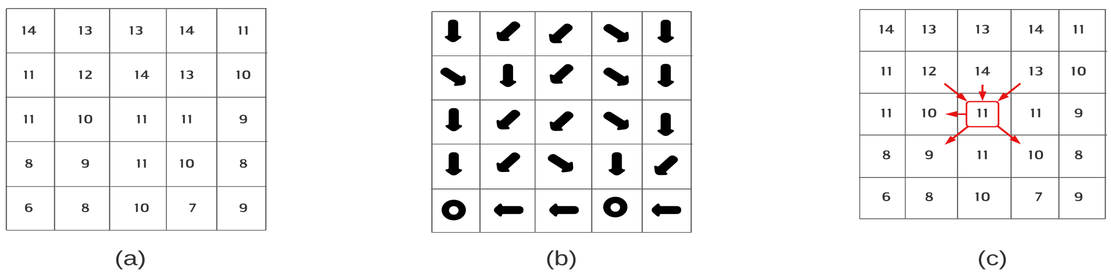


Figure 3.3: D8 flow algorithm example (a) Elevation exemplary data (b) D8 based flow directions (c) Single cell in-flow and out-flow.

cells or to stay in the same location, given as

$$a^{u_i} : \{N, S, E, W, NE, NW, SE, SW, hover\}$$

where, N is north, S is south, E is east and W is west movement. Each UAV chooses one action in a time-step.

The reward function encodes the objectives of the system into the policy. The UAVs aim to acquire a policy that maximizes the long-term accumulation of rewards. The reward function $R_t^{u_i}$ is formulated based on transition-derived information, represented as:

$$R_t^{u_i}(s_t^{u_i}, a_t^{u_i}, s_{t+1}^{u_i}) = I_{c,t_j}^{u_i} - \beta(s_{t+1}^{u_i}, U) \tag{3.4}$$

where, $I_{c,t_j}^{u_i}$ is the information gain as given by Equation 3.3. $\beta(s_{t+1}^{u_i}, U)$ returns a collision penalty if more than one UAV takes an action to arrive at the same state s_{t+1} at time $t+1$. U is the set of UAVs in the system.

3.3.2 Water Flow Direction Estimation

The standard D8 method [45] maps the direction of water flow for each grid cell to its steepest neighbour. Figure 3.3(a) and Figure 3.3(b) illustrates the D8 methodology. In the considered scenario, the input to the D8 model is the state vector (s^{u_i}) of the UAV u_i. The D8 method then generates a water flow direction (refer Figure 3.3(c), Figure 3.4(c)) which is mapped to a feasible action a^{u_i} for the UAV u_i.

Governing Equations

The distributed hydrological model for flowing water is generally based on the Saint Venant conditions [106] which are composed by the continuity and momentum equations, conditioned as following:

$$\frac{\partial h}{\partial t} + \frac{\partial \omega}{\partial c} = I_e \tag{3.5}$$

$$\frac{\partial \omega}{\partial t} + \frac{\partial}{\partial c}\left(\frac{\omega^2}{h}\right) - g.h\left(s_g - \frac{\partial h}{\partial c}\right) + g.h.s_f = 0 \tag{3.6}$$

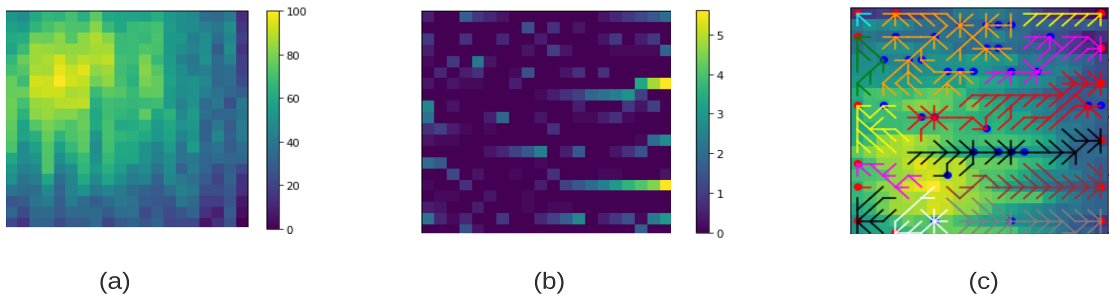


Figure 3.4: D8 flow algorithm in simulation (a) Elevation data of the region (b) Water accumulation in the region (c) D8 based flow directions.

where, h is the water depth. I_e is the intensity of excess rainfall (encoded using prior data of the region [107]). s_g is the ground slope and s_f is the friction slope. ω is the water discharge per unit/cell at the current time-step t. g is the value of gravity.
The discharge of water at each cell is derived by applying Manning's equation as following (for more details see [106, 108]):

$$\omega = \frac{h^{\frac{5}{3}}}{\phi} \tag{3.7}$$

where, ϕ is the Manning roughness coefficient.

Numerical Scheme

Based on the concept of the backward-finite-difference scheme, the continuity equations can be represented as (for more details see [106]):

$$h_{t+\Delta t}(c) = h_t(c) + (\omega_t^{in}(c) - \omega_t^{out}(c))\frac{\Delta t}{\Delta c} \tag{3.8}$$

where, $\omega_t^{in}(c)$ is the inward discharge for cell c at time t and $\omega_t^{out}(c)$ denotes the outward discharge, considering only the immediate neighbouring cells. The discharge of cell c at time-step $t + \Delta t$ is derived by applying the discretized momentum equation as following:

$$\omega_{t+\Delta t}(c) = \frac{(h_{t+\Delta t}(c))^{\frac{5}{3}}}{\phi} \tag{3.9}$$

$$L_{ct}^{u_i} = \underset{c_j \in c_k}{argmax}\left(\frac{\mathcal{X}(c^{u_i}, c_j).f_{rate}}{d(c^{u_i}, c_j)}\right) \tag{3.10}$$

$$\mathcal{X}(u_{c_i}, c_j) = (\omega_{t+\Delta t}(c^{u_i}) - \omega_{t+\Delta t}(c_j))\,. \tag{3.11}$$

where, c^{u_i} denotes the cell occupied by a UAV u_i. c_j denotes one of the neighbouring cells to the cell c^{u_i} from the set of 8 possible neighbours given by c_k. Distance between two cells $d(.,.)$ is calculated by finding the minimum number of cells between them. The diagonal cells next to each other have a distance of $\sqrt{2}$ between them and the horizontally/vertically aligned cells next to each other have a distance of 1 between them. $L_t(.,.)$ calculates the cell having the lowest relative discharge in the neighbourhood of u_i. The cell experiencing the lowest discharge tends to accumulate more water (refer Figure 3.4(b)), making it more likely to be considered the most critical cell within its surrounding neighborhood. The UAV u_i may also decide to hover at the same cell without moving if the current cell itself has the lowest discharge at time t. When a UAV has found a neighbouring cell with the lowest discharge, it moves one step in that direction and the procedure is repeated.

$$a_{c_t}^{D8} = \begin{cases} m(L_t(u_{c_i}), c^{u_i}) & if\ \mathcal{X}(u_{c_i}, c_j) > 0 \\ hover & otherwise \end{cases} \tag{3.12}$$

where, a_t^{D8} represents the D8 generated action for an individual UAV. The function $m(.,.)$ maps the correct action from the feasible action set (refer subsection 3.3.1). For example, let say the output of function $L_t(.,.)$ is cell c_2 and currently the UAV is at cell c_5, then output of function $m(.,.)$ is N denoting north (assuming a virtual 3x3 grid centered at c_5).

3.3.3 Directed Explorations: D8 flow based Action Selection

The action selection process in RL helps to manage the dual objectives of exploration and exploitation in the best possible manner. In RL, a state-action value function ($Q(s,a)$) is defined as a quantitative measure that judges how good an action is in any given state. Higher the Q-value, better is the action in that state. Most of RL algorithms use an ϵ-greedy exploration-exploitation strategy for the selection of actions during the training phase. The ϵ-greedy action selection method selects a random action with probability $\epsilon \in [0,1]$ for exploration in the state-space while it selects the action having maximum $Q(s,a)$ value with $1-\epsilon$ probability for the exploitation of the already acquired experience.

As this exploration is random, it leads to poor rewards in early stages. This form of exploration is even more detrimental when the task is time-sensitive as in the case of flooded area coverage. By using the governing equations and numerical scheme of water flow (refer subsection 3.3.2), the most probable action for exploration is estimated to accumulate higher information gain. Still some randomness is maintained in action selection to avoid over-fitting and learning more robust and generalized policies.

The proposed action selection strategy for directed explorations is given as:

$$a_t^{u_i} = \begin{cases} \underset{a'}{argmax}\ Q(s_t^{u_i}, a') & with\ 1-(\epsilon_1+\epsilon_2)\ probability \\ a_{c_t}^{D8} & with\ \epsilon_1\ probability \\ random\ \ action & with\ \epsilon_2\ probability \end{cases} \tag{3.13}$$

where, $0 \leq \epsilon_1 \leq 0.5$, $0 \leq \epsilon_2 \leq 0.5$. Further, the ϵ probabilities are decayed as training progresses with ϵ_2 decaying faster than ϵ_1. A shift from random actions to D8 based actions is made as the latter produces better action sequences which help in achieving the overall objective in an efficient and quicker manner.

3.3.4 Learning Action Values

The state-action value function $Q^{\pi}(s_t, a_t)$ under the policy π defines the long term desirability of an action in a particular state [19], given as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{a_t \sim \pi}\left[\sum_{l=0}^{\infty} \gamma^l R^{u_i}_{t+l+1} | s_t, a_t\right] \tag{3.14}$$

where $0 < \gamma < 1$ is the discount factor.

The proposed directed exploration strategy can be included in any Q-value based RL algorithm. Here, two algorithms are proposed, namely, D8-Q-Learning (D8QL) and D8-Deep-Q-Network (D8DQN) that implement the proposed action selection strategy in the baseline Q-learning [19] and DQN [13] algorithms, respectively.

The Q-value update equations in the proposed D8QL and D8DQN remain the same as their baseline versions with the change in the way states are explored during training.

Q-learning update equation is given as:

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(R^{u_i}_t + \gamma.\ \underset{a'}{max} Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{3.15}$$

DQN [13] uses two deep neural networks - Q network and target network with parameters θ and θ^- respectively. DQN loss for training the deep model is calculated as:

$$\mathcal{L}(\theta) = \mathbb{E}\left[(y - Q(s_t, a_t; \theta))^2\right] \tag{3.16}$$

$$y = R^{u_i}_t + \gamma \cdot \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \theta^-) \tag{3.17}$$

In D8QL, each UAV learns its own separate policy using replay buffer to query the stored information. As Q-learning is a tabular method that maintains a table entry for all the possible state action pairs, it doesn't scale well to large state spaces with increasing number of grid cells. Deep neural network based D8DQN can cater to complex large-scale state-spaces. In the purposed D8DQN, the experience of all the UAVs is stored in a single replay buffer that can be used to train the Q network for each UAV's policy from the joint experience. The collision-free multi-UAV movements and non-repeated explorations of the grid cell are ensured by the reward function (see Equation 3.4) during training.

The proposed D8DQN approach is depicted in algorithm 1. The action value functions Q^{u_i} and $\hat{Q}^{u_i}$ are initialized randomly for each UAV u_i. The algorithm executes episodes for each UAV adhering to specified limits on time-steps and energy consumption. During each episode, actions based on exploration rate probabilities ϵ_1 and ϵ_2 are performed or an action giving the maximum Q-value for the current state is selected. After executing an action, rewards are received and transition information is stored in replay buffer $\mathcal{Z}$. Mini-batches from $\mathcal{Z}$ are sampled and a gradient descent step is performed w.r.t. Equation 3.16. Periodically, the target action value function $\hat{Q}^{u_i}$ is updated to match the current action value function Q^{u_i}.

Algorithm 1: D8DQN Algorithm

```
1  Input: $\mathbb{H}$, $\gamma$, $\beta$, $\Delta\psi$, number of UAVs $n$
2  Initialize action value function $Q^{u_i}$ with random weights $\theta^{u_i}$ for each UAV $i \in U$
3  Initialize target action value function $\hat{Q}^{u_i}$ with weights $\theta^{-} = \theta^{u_i}$ for each UAV
     $i \in U$
4  for $UAV = u_i, u_j, ..., u_n$ do
5      for episode=1,2,... do
6          Initial observation $s^{u_i}$
7          while $t \leq max\_time\_step$ and $\psi_t > \Delta\psi$ do
8              With probabilities $\epsilon_1$ and $\epsilon_2$ select exploratory action $a_t^{u_i}$ or otherwise
                 select max. Q-value action, as per:
9              $$a_t^{u_i} = \begin{cases} \underset{a'}{argmax}\ Q(s_t^{u_i}, a') & with\ 1-(\epsilon_1+\epsilon_2)\ probability \\ a_{c_t}^{D8} & with\ \epsilon_1\ probability \\ random\ \ action & with\ \epsilon_2\ probability \end{cases}$$
10             Execute action $a_t^{u_i}$
11             UAV $u_i$ makes the next observation $s_{t+1}^{u_i}$ and receives reward $R_t^{u_i}$ from
                 the environment.
12             Store transition $< s_t^{u_i}, a_t^{u_i}, R_t^{u_i}, s_{t+1}^{u_i} >$ in replay buffer $\mathcal{Z}$
13             Sample a random mini-batch of $\mathcal{B}$ transitions $(s_k^{u_i}, a_k^{u_i}, R_k^{u_i}, s_{k+1}^{u_i})$ from
                 $\mathcal{Z}$
14             Calculate: $y_k^{u_i} = R_k^{u_i} + \gamma \cdot \max_{a'} \hat{Q}^{u_i}(s_{k+1}, a'; \theta^{-})$
15             Perform gradient decent step on $(y_k^{u_i} - Q(s_k^{u_i}, a_k^{u_i}; \theta))$ w.r.t. network
                 parameter $\theta$
16             In every $C$ steps, reset $\hat{Q}^{u_i} = Q^{u_i}$
17         end
18     end
19 end
```

3.4 Experimentation and Results

In this section, the performances of the proposed algorithms D8QL for discrete state-space and D8DQN for continuous and/or large state-space is evaluated. The following baselines and state-of-the-art techniques are considered for comparison, namely, Q-Learning (QL), DQN, Random Walk (RW), Genetic Algorithm (GA) and A* algorithm. The difference between the baseline QL and the proposed D8QL is the directed exploration using D8 flow algorithm. A simple baseline of randomized actions is also implemented to analyze how other algorithms perform. Authors in [109] implement a mobility model for a convoy of UAVs. The baseline mobility model employed in [109] is called Random Mobility Model where each UAV chooses its next orientation based on the output of a random value function. The Random Walk (RW) model is applied in reference to this Random Mobility Model for implementation. Another algorithm used for comparison is Genetic algorithm (GA) [110] in reference to the literature. In [110], authors try to optimize the path of the UAV by formulating it as an optimization problem for energy consumption. They address this issue by applying GA as an optimizer in their model. The A* algorithm is implemented based on [111]. The authors in [111] suggest a greedy A* technique as an online coverage approach for multi-robot systems. This approach assists the robots in moving closer to the unvisited regions.

The elevation grid over Chennai City of India is used to generate the training environment (this city had a devastating flood in December 2015 and is seen as a flood prone region upto date). Different inundated areas corresponding to different water levels are generated over the elevation grid using Mapbox API [112]. There are 8 levels that define the water level in each of grid cell. These levels are: $\{1ft, 2ft, 3ft, 4ft, 5ft, 6ft, 7ft, 8ft\}$. The data corresponding to population density is collected using World flood mapping tool [113]. There are 5 levels of population density ranging from $\mathbb{P}_1$ to $\mathbb{P}_5$. $\mathbb{P}_1$ denotes population density of 5 to 10 people in 100 square meter area, $\mathbb{P}_2$ denotes population density of 10 to 25 people, $\mathbb{P}_3$ denotes population density of 25 to 50 people, $\mathbb{P}_4$ denotes population density of 50 to 100 people and $\mathbb{P}_5$ that denotes population density of more than 100 people in 100 square meter area. For each experiment, the UAVs are randomly initialized over different grid cells. A ground control station is assumed to be in place that communicates with the UAVs to process the collected data. Implementation is done on Google Colab having, Intel(R) Xeon(R) CPU, 1xTesla K80 GPU, 2.30GHz CPU frequency, and 12GB RAM.

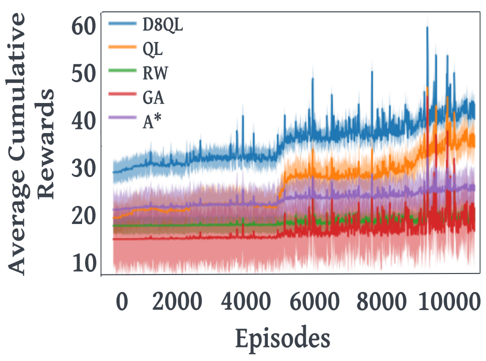
As there is no communication among the UAVs, each UAV operates independently of the others. The rewards accumulated by the UAVs is used as a performance metric. Additionally, the joint area coverage and the number of collisions among the UAVs during training are analyzed. The policies trained using the proposed algorithms are also evaluated in an unseen test environment.

3.4.1 Performance in Small Discrete State-Space

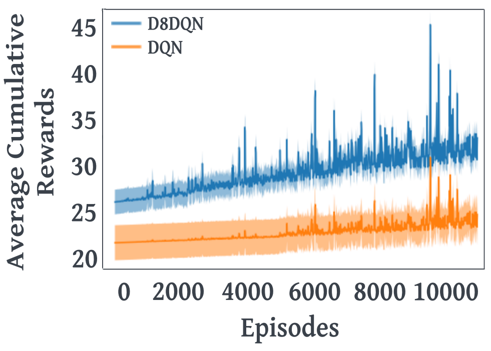
Experiments were performed over the course of 10000 episodes, where each episode was of 1000 time-steps. Average cumulative reward (per UAV) in each episode has been recorded for comparison. A total of 5 UAVs were used for this experiment. The altitude of the UAVs was fixed at 50 meters above sea level. Considering a standard UAV camera, having half angles of 30° and 45°, results in an FoV coverage of approximately 6000 m^2. Considering a small sub-region of Chennai city area of 2.5x10^6 m^2 approximately, a grid size of 20 x 20 is possible where each cell is equal to the size of the UAV's FoV.

With each episode, the latitude and longitude is slightly shifted to gather evaluation data of nearby regions in order to generalize the trend of water flow in the broader region. The rate of water flow is kept constant at 2 m/s throughout the episodes and its intensity changes based on the ratio of elevation of the considered grid cells. Each cell has underlying real-world data of elevation values, encoded using Mapbox API [114]. Using Mapbox.js, a terrain DEM (digitally elevated model) is created. By overlaying the UAV captured FoV on top of the DEM layer, the data behind the map is queried using the Surface API.

As can be seen from Figure 3.5a, RW is not able to evolve its performance with the increasing number of training episodes. This corresponds to the lack of learning capability of RW that only uses randomized action function. The GA method shows some gain



(a)



(b)

Figure 3.5: (a) Average Cumulative Rewards observed by RW, GA, A*, QL and D8QL over a small discrete state-space. (b) Average Cumulative Rewards observed by DQN and D8DQN in a city scale flooded region.

in rewards after 5000 episodes, however it's still very low in comparison to Q-learning methods. This highlights the effectiveness of reinforcement learning methods when applied to dynamic and unknown environments. A* performs at par with Q-learning in early episodes but is unable to accumulate more rewards in later episodes. Q-learning method shows an impressive jump in rewards around 5000^{th} episode and shows significant rise in cumulative rewards after that. This significant change highlights the effectiveness of the exploration-exploitation strategy (ϵ-greedy) of Q-Learning. The proposed D8QL outperforms all and is able to accumulate maximum rewards from the initial episode onwards. This high accumulation of rewards especially in the initial episodes corresponds to directed explorations based on the D8 flow estimates within the Q-learning algorithm that helps in better action selection than $\epsilon-$greedy strategy.

3.4.2 Performance in Large State-Space

Experiments are conducted over a larger grid size of 260x260 cells considering the complete area of Chennai city of approximately 425 x10^6 m^2, with the same altitude for the UAVs as before. As Q-table becomes too large to accommodate the whole state-action space, the proposed D8DQN algorithm is implemented in this scenario. This technique is compared with the baseline DQN method [28] to analyze the effect of directed explorations using D8 flow estimates. As can be seen in Figure 3.5b, D8DQN and DQN both shows similar trend of rise in rewards as episodes go by. However, D8DQN is able to acquire higher cumulative rewards from first episode itself and is also able to achieve highest rewards by 9000^{th} episode. This highlights the fact that the multi-UAV trained using D8DQN is able to perform non-repeated observations of the grid cell with high information gain in a better fashion. Hence, the directed explorations induced by the D8 flow estimates help in an improved performance of D8DQN method as compared to its baseline.

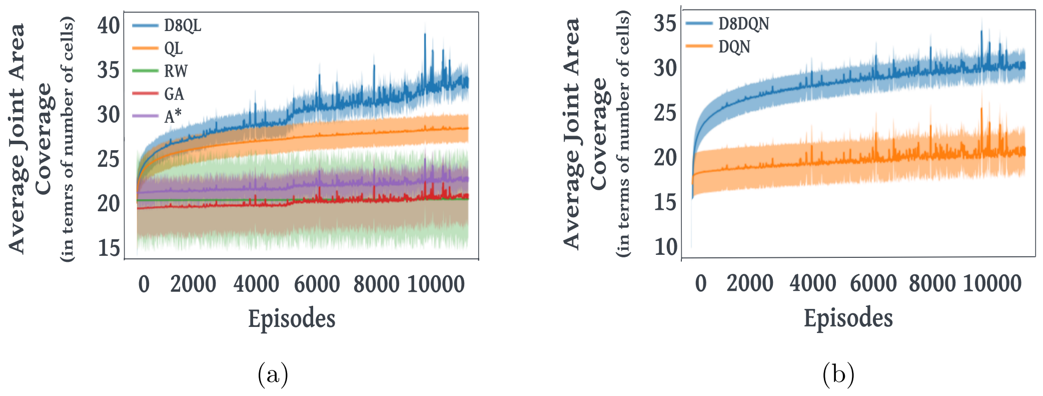


Figure 3.6: (a) Performance comparison between RW, GA, A*, QL and D8QL over the number of cells covered. (b) Performance comparison between D8DQN and DQN over the number of cells covered.

3.4.3 Comparing Cell Coverage during Training

In this experiment, the coverage of grid cells observed by various algorithms is analyzed. The grid cell coverage is defined as the number of cells covered by the UAVs until their battery is completely drained. A total of 5 UAVs with 50 energy units each are considered in this experiment. A linear energy depletion rate is considered which implies, at maximum 50 cells can be covered by a UAV in a single episode if in each time-step it decides to change its location. Figure 3.6a depicts the performance observed in smaller discrete state-space among D8QL, QL, RW, GA and A*. As observed, GA and A* sees a similar rise in area coverage throughout the episodes with A* performing better. GA is eventually able to outperform RW around 5000^{th} episode. QL and D8QL both show logarithmic rise in area coverage that somewhat flattens at around the 10000^{th} episode.

Figure 3.6b depicts the performance of DQN and D8DQN for cell coverage over large city scale state-space. Similar to D8QL, D8DQN also shows a logarithmic pattern in terms of coverage that flattens at around 10000^{th} episode. DQN performs rather poorly with no considerable rise in area coverage. This signifies that area and critical region coverage are not equivalent and even when DQN doesn't observe higher area coverage it still is able to gather significant rewards by covering the critical regions.

3.4.4 Number of Collisions Observed during Training

Another metric considered to analyze the performance of the proposed algorithms is the number of collisions observed by each algorithm during training. As per Equation 3.4, each UAV receives a penalty whenever two or more UAVs are in the same cell. This experiment helps in analyzing whether the proposed techniques learn to avoid these penalties in the long run or not. From Figure 3.7a, it can be observed that RW makes close to zero progress in reducing the number of collisions with increasing number of episodes. Similar performance is observed in case of GA. This highlights that RL based methods learn to avoid these penalties in more constructive manner as compared to other methods. Still, A* is able to perform much better than GA and shows similar pattern as compared

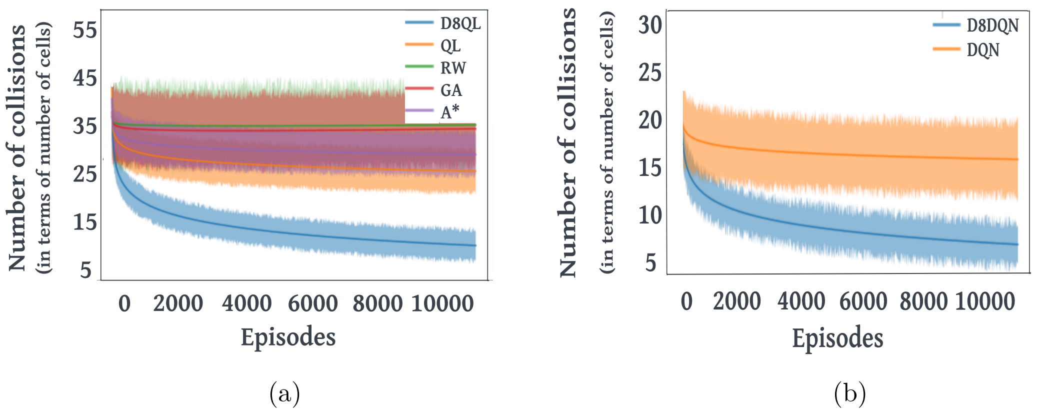


Figure 3.7: (a) Number of collisions observed by RW, GA, A*, QL and D8QL over the duration of training period. (b) Number of collisions observed by D8DQN and DQN over the duration of training period.

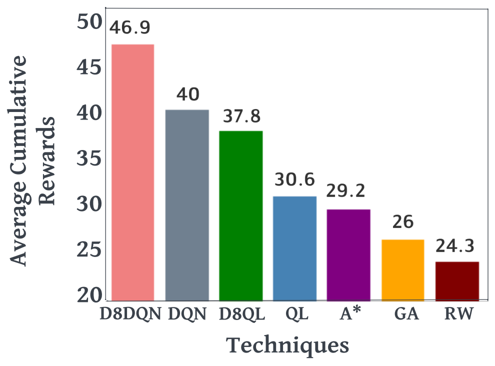
to Q-learning in terms of fall in number of collisions with the increasing number of episodes. However, as can be observed, the D8QL algorithm outperforms all other methods showcasing a better multi-UAV policy. Considering the neural network based techniques for large state-spaces i.e. DQN and D8DQN, D8DQN performs significantly better than DQN in number of collisions as can be seen in Figure 3.6b.

3.4.5 Policy Generalization in Unseen Test Environment

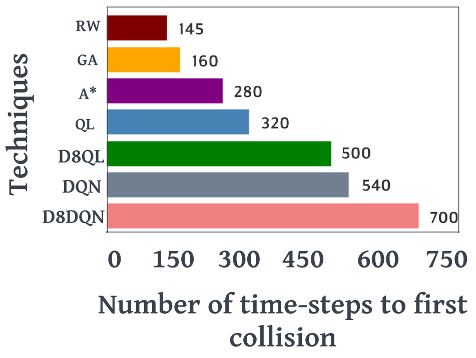
Both of the proposed methods D8QL and D8DQN are tested against other baselines in a new unseen environment to evaluate the generalizability of the proposed approaches for multi-UAV based area coverage. A similar coastal region of approximately 2.5x10^6 m^2 area is selected (from Mumbai city coastal region) having roughly 20 x 20 size grid, where each cell is equal to the size of the UAV's FoV. Figure 3.8a depicts the average cumulative rewards accumulated by different methods in the unseen test region over a single episode with 1000 time-steps. It can observed that the D8DQN model is able to generalize its performance better in the unseen test environment as compared to the other methods. This can be attributed to the fact that Deep-Q-Network based technique learns by adjusting the weights and biases in a deep neural network in order to approximate the best policy. The D8QL is still able to generalize better than other methods due to the robust state representation having the elevation information of the neighbouring cells. Standard DQN slightly outperforms D8QL highlighting the effectiveness of Deep RL algorithms in unseen and unknown environments.

3.4.6 Time-step to First Collision in Unseen Test Environment

Further, observations are recorded regarding the duration after which the multi-UAV system encounters their first collision. This is observed in the unseen test region which is performed over a single episode with 1000 time-steps. Figure 3.8b highlights these results and provides with a similar order of performance as observed in other experiments.



(a)



(b)

Figure 3.8: (a) Average Cumulative Rewards observed by RW, GA, A*, QL, D8QL, DQN and D8DQN in an unseen test region. (b) Comparison between RW, GA, A*, QL, D8QL, DQN and D8DQN over the number of time-steps to record first collision.

D8QL, DQN and D8DQN perform significantly well as compared to RW, GA, A* and QL techniques. As higher penalties depict more number of collisions observed by the system, this result highlights that D8QL, DQN and D8DQN are relatively better than other techniques in handling penalties observed in the environment.

3.5 Chapter Summary

This chapter introduces D8QL and D8DQN algorithms, for autonomous multi-UAV control to perform area coverage during floods. A novel exploration strategy using D8 water flow algorithm is proposed to expedite the training process. Both the algorithms D8QL and D8DQN depict significant improvement in terms of rewards, area coverage, and UAV collision avoidance, as compared to other techniques discussed in the literature. However, limitations exist concerning discrete action spaces and UAV clustering, resulting in stiff UAV motion and less effective coverage of critical regions. The clustering problem arises from the D8 algorithm, causing all the UAVs to converge in close proximity to one another, thereby hindering coverage and increasing the risk of collisions. The next chapter thus focuses on exploring techniques for continuous UAV actions in flooded environments and mitigating the clustering effect.

4| Continuous Multi-UAV Control with Directed Explorations during Floods

This chapter extends the previous work presented in chapter 3 on Deep RL based multi-UAV area coverage during floods. In this chapter, a continuous action space algorithm is proposed for learning the UAV control policies for smoother motion. The proposed approach addresses the limitations of the previous work, which uses Deep Q Networks (DQN) having discrete action set and an exploration strategy that leads to clustered formations of UAVs. In this work, the Deep Deterministic Policy Gradient (DDPG) algorithm with an improved target actor is presented, incorporating domain knowledge using the D-infinity (DINF) algorithm. The water flow estimates obtained from DINF are used to guide the UAVs' actions especially in the early phases of training. In addition, Path scatter is introduced to inhibit clustered formations and maintain inter-UAV separation for better area coverage. The objective remain the same, i.e., to identify densely populated areas that are susceptible to a large volume of water accumulation. The task is time-sensitive in nature due to the limited battery of the UAVs.

The rest of the chapter is organized as follows. section 4.1 contains the preliminary information highlighting the environment description and the basics of the DINF flow estimation algorithm. section 4.2 describes the multi-UAV system with collision avoidance and UAV energy model. In section 4.3, the proposed Deep RL approach is presented with a novel target actor based on DINF. section 4.4 discusses the experiments and results. Finally, in section 4.5, a concise summary of the chapter is provided.

4.1 Preliminaries

In this section, the flood environment and its characteristics are discussed. Additionally, the underlying idea behind the functioning of the DINF algorithm is presented.

4.1.1 Environment Description

The environment consists of a 2D terrain, similar to the one considered in chapter 3, section 3.1. Real-world data corresponding to the elevation levels of the terrain is encoded while rendering the environment using Mapbox Terrain-DEM [114]. A 2D mask representing the water layer is defined over the terrain to simulate floods. There are two parameters that define the surface water, one being the water level h and the other being the speed of the water current f_{rate}. The population density levels ($\mathbb{P}$) are encoded based on the regional map that is generated using the World Flood Mapping tool [113], depicting the varied population levels over a 100x100 m^2 area.

The population density level along with the water level defines a Critical level ($\mathbb{Z}$) of a location c given as:

$$\mathbb{Z}_c = h_c \times \mathbb{P}_c \tag{4.1}$$

Critical level $\mathbb{Z}_c$ of a sensed location c, defines the risk at that particular location highlighting the high population density regions that are under an immediate threat of large accumulation of flood water. Both h_c and $\mathbb{P}_c$ are discretized over a range to give a finite number of critical levels.

4.1.2 D-Infinity based Water Flow Estimation

The working of the D-Infinity (DINF) algorithm is based on the hydrological model for flowing water. It is generally based on the Saint Venant conditions [108] which are composed of the continuity and momentum equations (for more details refer to chapter 3, subsection 3.3.2). DINF flow estimation [115] algorithm, determines the direction of water flow based on the slope of triangular facets geometrized over the environment (centred at the location of interest). This flow-partition approach uses elevation information (i.e., encoded in the environment) to determine the fractional flow of water draining to the downslope neighbouring locations. DINF is a multi-flow direction (MFD) algorithm, that provides: (1) continuous flow angles, and (2) flow partitioning between the neighbouring locations. The output of DINF is set to a floating point value depicting an angle in degrees going counter-clockwise from 0° to 360°. The water accumulation at a particular location is the aggregation of its current water level and the contribution from upslope neighbouring locations. The flow from each location either contributes to just one neighbour (when all the other neighbours are at a higher elevation) or is on an angle falling between the direct angle of two adjacent neighbours.

The location with the highest water accumulation (denoted as $L^{u_i}_{c_t}$) in the neighbourhood of c^{u_i} (where c^{u_i} is the location that is currently being sensed by the UAV u_i,) is calculated as (for more details refer to subsection 3.3.2):

$$L^{u_i}_{c_t} = max\left(\frac{\mathcal{X}(c^{u_i}, c_j).f_{rate}}{d(c^{u_i}, c_j)}\right) \ \forall\ j \in c_k \tag{4.2}$$

$$\mathcal{X}(u_{c_i}, c_j) = \left(\omega_{t+\Delta t}(c^{u_i}) - \omega_{t+\Delta t}(c_j)\right). \tag{4.3}$$

where, t denotes the time-step and c_k represents the 8 neighbouring regions surrounding location c^{u_i}. $d(.,.)$ is the Euclidean distance between two locations and $\omega_{t+\Delta t}(c^{u_i})$, $\omega_{t+\Delta t}(c_j)$ denotes the water discharge at location c^{u_i}, c_j respectively.

In the subsequent section system model is described.

4.2 System Model

In the system model, there is group of n quadrotor UAVs$(U) = \{u_i | i \in 1, 2, .., n\}$ that are tasked to perform optimal area coverage of an unknown flooded region. The task is time-sensitive due to UAV's limited battery. As the dynamics of the environment are unknown, the UAV's actions are realized using a policy gradient based Deep RL algorithm, known as, Deep Deterministic Policy Gradient (DDPG) [32]. The altitude of the UAVs is fixed at $\mathbb{H}$ meters above the ground to have the same resolution images from all the UAVs for further processing, such as survivor detection, image mosaicking, etc. Each UAV is equipped with ventral cameras that provide a rectangular field of view (FoV). The FoV of each UAV forms a pyramid having half angles (θ_1,θ_2) as shown in Figure 4.1. In the group of n UAVs, each UAV acts individually with distinctive control policies. Each UAV is assumed to have Full-duplex communication with a ground control station that handles the processing overload.

Location Specific Information Gain

The information gain from a location is defined based on the critical level ($\mathbb{Z}$) of that location w.r.t. the environment, as defined in Chapter 3, section 3.2. The UAVs are rewarded highly if the information gain from a particular location is high at the observation time. The actual information gain ($I^{u_i}_{c,t_j}$) for a given location c sensed by UAV u_i is calculated as:

$$I^{u_i}_{c,t_j} = \frac{\mathbb{Z}_c}{max(\mathbb{Z})} \times \frac{t_j - t_i}{t_j} \tag{4.4}$$

where, t_i denotes the last time-step at which the location c was observed and t_j denotes the current time-step. The primary aim of introducing information gain is to prompt the UAVs to capture more unobserved regions/locations rather than re-visiting the observed ones. The observed location is the one that has already been sensed by a UAV recently.

4.2.1 Multi-UAV Collision Avoidance

Given the multi-UAV setup, a collision avoidance protocol has been established among the UAVs, incorporating a system penalty to address potential collisions. A collision ($\mathbb{C}$) is recorded whenever two or more UAVs violate the separation threshold λD ($\lambda < 1$), where D is the minimum separation between the UAVs to avoid overlap.

$$\mathbb{C}_t(u_i, u_j) = \begin{cases} \beta \quad if \;\; d^{u_i,u_j}_t < \lambda D \quad \forall i, j \in n \\ 0 \qquad\qquad otherwise \end{cases} \tag{4.5}$$

where β is a scalar denoting a collision penalty and λD is the minimum separation required between the UAVs such that they do not incur collision penalties, as seen in Figure 4.1.

$$d^{u_i,u_j}_t = \sum_{i,j \in U, i \neq j} \sqrt{(x^{u_i} - x^{u_j})^2 + (y^{u_i} - y^{u_j})^2} \tag{4.6}$$

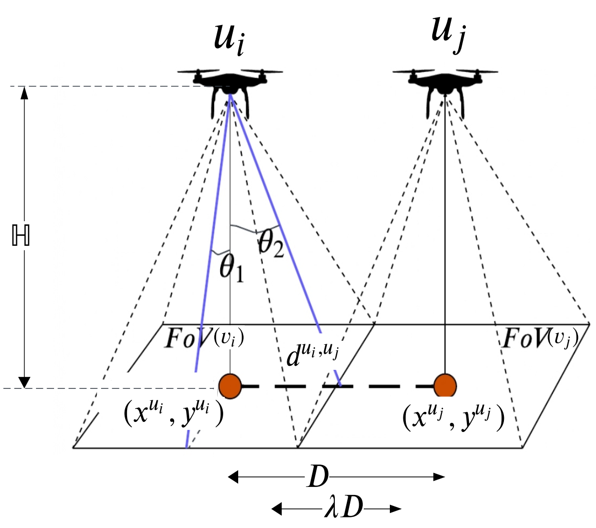


Figure 4.1: Two UAVs u_i and u_j maintaining some threshold distance to avoid collision penalty.

where $d_t^{u_i,u_j}$ is the Euclidean distance between the 2D projected positions of the UAVs u_i and u_j as seen in Figure 4.1. As the UAV positions are defined in latitude and longitude coordinates, the output provided by the function $d(.,.)$ is expressed in degrees. To convert this angular measurement to meters, the output should be multiplied by the approximate factor of 1.11×10^5, where each degree corresponds to approximately this distance in meters [116].

4.2.2 Energy Depletion

The flight time of a UAV is restricted due to its limited battery. Expanding upon the energy model introduced in Chapter 3, section 3.2, the consideration involves the rate at which energy depletes for the UAV based on whether it is in motion or stationary while hovering. The energy depletion for a UAV is directly proportional to the number of time-steps during which the UAV has been in flight. The UAV's energy ψ_t at any given time t is given as:

$$\psi_t = \begin{cases} \psi_{t-1} - \Delta d_1 & if \;\; action = hover \\ \\ \psi_{t-1} - \Delta d_2 & otherwise \end{cases} \tag{4.7}$$

where, Δd_1 and Δd_2 are two positive scalars, where $\Delta d_2 > \Delta d_1$. The energy depletion is higher in the case when the UAV is moving as to when it's hovering [117]. Energy affects the flight time of the UAV based on the UAV's action at each time-step however, energy is not an intrinsic part of the objective as an optimization problem.

4.2.3 Expanding Coverage with Path scatter

To broaden the coverage of UAVs over the environment, Path scatter ($\mathcal{O}$) is introduced for the model that impacts the model's reward formulation. In some scenarios, the starting configuration of the UAVs' or their actions may lead to a clustered formation. To mitigate that, the Path scatter helps the UAVs to maintain some minimum dispersion over the region. The Path scatter is calculated based on the distance between the UAVs at time-step t (refer to Equation 4.6) and is given as:

$$\mathcal{O}_t(u_i, u_j) = \begin{cases} 1 & if \ \ d_t^{u_i,u_j} \geq 2D \\ \delta & if \ \ D \leq d_t^{u_i,u_j} < 2D \\ \xi & if \ \ \lambda D \leq d_t^{u_i,u_j} < D \end{cases} \tag{4.8}$$

where $0 < \xi < \delta < 1$ and $d_t^{u_i,u_j}$ is the pairwise Euclidean distance between the 2D projected positions of the UAVs $u_i, u_j \forall i, j \in U, i \neq j$.

4.3 Proposed Methodology

In this section, the considered multi-UAV area coverage problem is discussed as an MDP, along with the DINF based control policy for the multi-UAV system.

4.3.1 MDP Formulation

An RL model can be described using a Markov Decision Process (MDP) quintuple: $< S, A, P_{ss'}, R, s_0 >$, where S represents the set of states. A is the feasible set of actions that a UAV can execute in a given state. $P_{ss'}$ represents the model of the environment, R is the reward function and s_0 is the starting configuration of the UAVs in the environment.

State

A state s^{u_i} of i^{th} UAV is represented as:

$$s^{u_i} : \{\mathbb{P}^{u_i}_{c+k1,c+k2}, e^{u_i}_{c+k_1,c+k_1}, h_c | \forall k1, k2 \in \{-1, 0, 1\}\}$$

where, c is the current location of UAV u_i. $\mathbb{P}_c$ denotes the population density level of a location c and e_c represents the terrain elevation of that location. h_c is the sensed water level at location c. The set $\{-1, 0, 1\}$ denotes the elevation and population density of eight neighbouring locations to location c. The water level of the neighbouring locations is not available *a priori* and hence it is not a part of the observable state.

Actions

The feasible action set for a UAV u_i is given as $A : \{pitch^{u_i}, yaw^{u_i}, hover^{u_i}\}$. Figure 4.2 depict these actions as angular rotations across different axes. The $pitch^{u_i}$ helps the UAV in forward and backward movement. The yaw^{u_i} rotates the UAV along the vertical axis which helps the UAV to change its heading direction. The $pitch^{u_i}$ value lies within the range of [-45°,45°], where 0° depicts the parallel alignment of the UAV with the horizontal

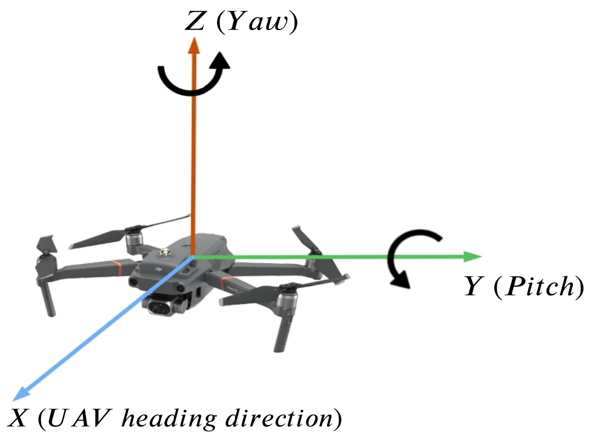


Figure 4.2: Yaw and Pitch rotations/actions of a Quadrotor UAV.

axis. A value less than 0° tilts the UAV where the head goes up and moves the UAV in a backward direction. A positive value of $pitch^{u_i}$ means that the head goes down and the UAV moves in the forward direction. yaw^{u_i} lies in the range [-180°,180°], where 0° corresponds to the head of the UAV in the north direction. A value lower than 0° means shifting the head of the UAV u_i in an anti-clockwise direction while a positive value of yaw^{u_i} means shifting the UAV head in a clockwise direction.

Rewards

The reward function encodes the objectives of the system into policy learning. The aim of UAV is to learn a policy that maximizes the long-term accumulation of rewards. The reward function $R_t^{u_i}$ is formulated based on the information obtained from a transition given as:

$$R_t^{u_i}(s_t^{u_i}, a_t^{u_i}, s_{t+1}^{u_i}) = I_{c,t_j}^{u_i}.\mathcal{O}(u_i, U) - \mathbb{C}_t(u_i, U) \tag{4.9}$$

where, $I_{c,t_j}^{u_i}$ is the information gain as given by Equation 4.4. $\mathcal{O}$ is the value of Path scatter as given by Equation 4.8. $\mathbb{C}_t(u_i, U)$ returns a positive scalar value corresponding to collision penalty (see Equation 4.5). The reward function represents the objective of the system in a quantitative manner, where the aim is to maximize this quantity in the long run by learning an optimal policy. Further to compare and improve policies in RL, the Q-value function $Q^\pi(s_t, a_t)$ defines the long-term desirability of a state-action pair as the expectation of the discounted cumulative sum of rewards, given as:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{a_t \sim \pi}\left[\sum_{l=0}^{\infty} \gamma^l R_{t+l+1}^{u_i} | s_t, a_t\right] \tag{4.10}$$

where $0 \leq \gamma < 1$ is the discount factor.

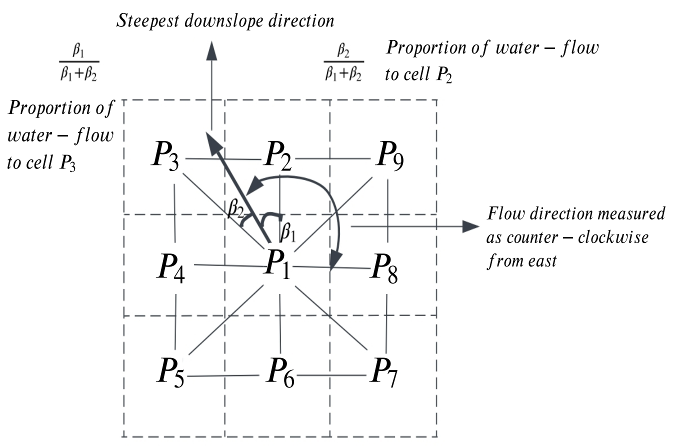

Steepest downslope direction
$\frac{\beta_1}{\beta_1+\beta_2}$
$\frac{\beta_2}{\beta_1+\beta_2}$
Proportion of water − flow to cell P_2
Proportion of water − flow to cell P_3
Flow direction measured as counter − clockwise from east
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
β_1
β_2

Figure 4.3: Working of D-infinity algorithm over a 3x3 grid space, centered at location P_1.

4.3.2 DINF based Actions

As UAVs have a continuous action space, knowing the exact degree of orientation is crucial to shifting the UAV's head such that it moves in the direction of the steepest water flow. In case of fractional discharge (as depicted in Figure 4.3), a triangular facet is formed with three vertices. The first vertex $P_1(x_{c_i}^{u_i}, y_{c_i}^{u_i}, e_{c_i}^{u_i})$ is the one where the UAV u_i is currently present. The second vertex $P_2(x_{c_j}, y_{c_j}, e_{c_j})$ is the location with the lowest relative discharge in the u_i's neighbourhood n^{u_i} (calculated using Equation 4.2). The third vertex $P_3(x_{c_k}, y_{c_k}, e_{c_k})$ is an adjacent location to the second vertex with relatively lower discharge. If the slope vector (see Figure 4.3 the steepest downslope direction) is outside the facet, the steepest flow direction is taken along the steepest edge. To determine the DINF based action, first the slope (downward) vectors (a_1, a_2) are calculated, as:

$$a_1 = \frac{(e_{c_i}^{u_i} - e_{c_j})}{\sqrt{(x_{c_i}^{u_i} - x_{c_j})^2 + (y_{c_i}^{u_i} - y_{c_j})^2}} \tag{4.11}$$

$$a_2 = \frac{(e_{c_j} - e_{c_k})}{\sqrt{(x_{c_j} - x_{c_k})^2 + (y_{c_j} - y_{c_k})^2}} \tag{4.12}$$

The aspect (α_c) of the facet w.r.t. the current cell c can then be derived as:

$$\alpha_c = tan^{-1}\left(\frac{a2}{a1}\right) \tag{4.13}$$

If the slope vector is outside the facet:

$$\alpha_c = \begin{cases} 0 & if \;\; \alpha_c < 0 \\ \Lambda & if \;\; \alpha_c > \Lambda \end{cases} \tag{4.14}$$

$$\Lambda = tan^{-1}\left(\frac{\sqrt{(x_{c_j} - x_{c_k})^2 + (y_{c_j} - y_{c_k})^2}}{\sqrt{(x_{c_i}^{u_i} - x_{c_j})^2 + (y_{c_i}^{u_i} - y_{c_j})^2}}\right) \tag{4.15}$$

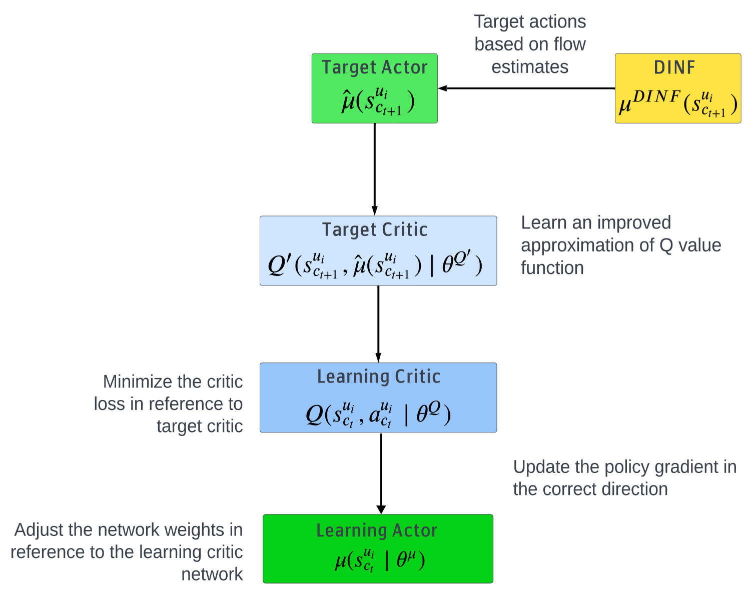

Target actions based on flow estimates
Target Actor
$\hat{\mu}(s^{u_i}_{c_{t+1}})$
DINF
$\mu^{DINF}(s^{u_i}_{c_{t+1}})$
Target Critic
$Q'(s^{u_i}_{c_{t+1}}, \hat{\mu}(s^{u_i}_{c_{t+1}}) \mid \theta^{Q'})$
Learn an improved approximation of Q value function
Minimize the critic loss in reference to target critic
Learning Critic
$Q(s^{u_i}_{c_t}, a^{u_i}_{c_t} \mid \theta^{Q})$
Update the policy gradient in the correct direction
Adjust the network weights in reference to the learning critic network
Learning Actor
$\mu(s^{u_i}_{c_t} \mid \theta^{\mu})$

Figure 4.4: Illustrating actor-critic architecture of DINF based DDPG.

Finally, a DINF policy μ^{DINF} for the i^{th} UAV can be given as:

$$\mu^{DINF}(s^{u_i}_{c_t}) = \alpha_c \tag{4.16}$$

where c is the cell occupied by the UAV u_i at time t.

4.3.3 Learning the Multi-UAV Policy

For learning a control policy for the UAV, a policy gradient based Deep RL algorithm is employed, known as Deep Deterministic Policy Gradient (DDPG), which is well suited for problems with continuous state-action spaces [32]. However, it has been reported in the literature that DDPG is susceptible to poor learning if significant rewards are not received in the initial phases of policy exploration [118]. In order to mitigate this problem, DINF based target actor is proposed to achieve directed explorations during early phases of training. The idea is to accumulate high rewards in the initial episodes itself for boosting the policy gradient in the correct direction. The effect of DINF is eventually diminished using ϵ decay as the aim is to learn a more robust and generalized policy.

Figure 4.4 depicts the flow of influence of the DINF exploration over the DDPG architecture. The DINF based Deep Deterministic Policy Gradient with Path scatter (D3S) model consisting of actor $\mu(s^{u_i}_t|\theta^{\mu})$ (or simply written as $a^{u_i}_t$) and critic $Q(s^{u_i}_t, a^{u_i}_t|\theta^{Q})$ networks and also have target networks for both actor $\hat{\mu}(s^{u_i}_{t+1})$ and critic $Q'(s^{u_i}_{t+1}, \hat{\mu}(s^{u_i}_{t+1})|\theta^{Q'})$. DINF based target actions are used to realize a better policy by affecting the target actor which in turn improves the critic network by minimizing the critic loss. The enhanced critic contributes to improving the actor network's training process.

The loss function used to update the critic network in the proposed model in reference to

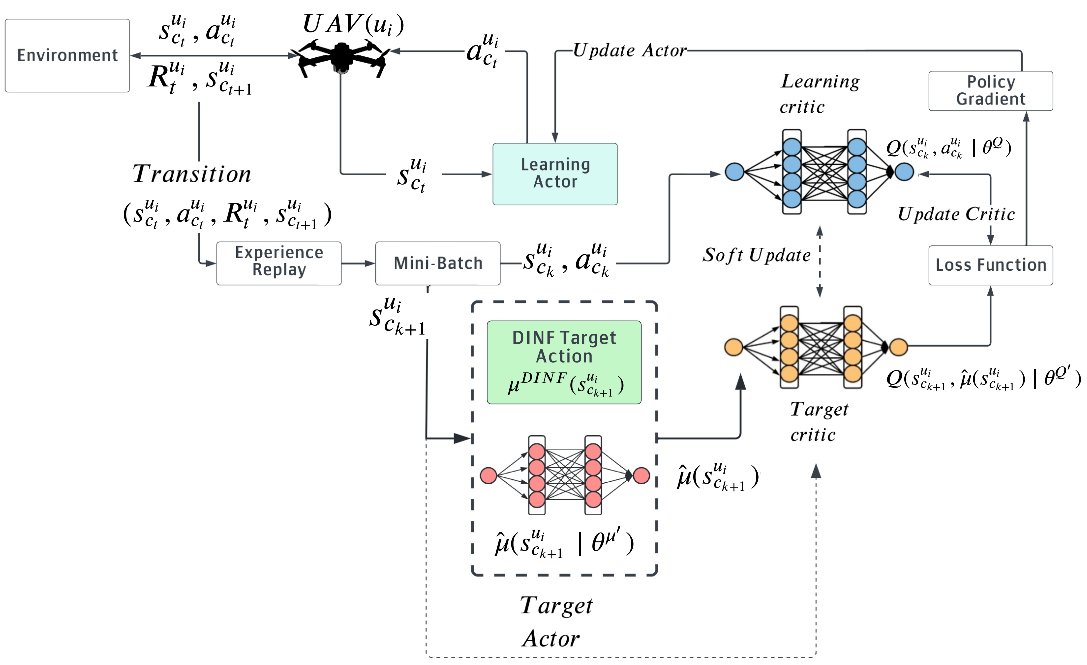

Environment
$s^{u_i}_{c_t}, a^{u_i}_{c_t}$
$R^{u_i}_t, s^{u_i}_{c_{t+1}}$
$UAV(u_i)$
$a^{u_i}_{c_t}$
Update Actor
Learning critic
Policy Gradient
Transition
$(s^{u_i}_{c_t}, a^{u_i}_{c_t}, R^{u_i}_t, s^{u_i}_{c_{t+1}})$
$s^{u_i}_{c_t}$
Learning Actor
$Q(s^{u_i}_{c_k}, a^{u_i}_{c_k} \mid \theta^Q)$
Update Critic
Experience Replay
Mini-Batch
$s^{u_i}_{c_k}, a^{u_i}_{c_k}$
Soft Update
Loss Function
$s^{u_i}_{c_{k+1}}$
DINF Target Action
$\mu^{DINF}(s^{u_i}_{c_{k+1}})$
$Q(s^{u_i}_{c_{k+1}}, \hat{\mu}(s^{u_i}_{c_{k+1}}) \mid \theta^{Q'})$
Target critic
$\hat{\mu}(s^{u_i}_{c_{k+1}})$
$\hat{\mu}(s^{u_i}_{c_{k+1}} \mid \theta^{\mu'})$
Target Actor

Figure 4.5: Illustrating proposed architecture of D3S algorithm.

UAV u_i currently at location c, is given as:

$$L_{critic} = \frac{1}{\mathcal{B}} \sum_t (y^{u_i}_{c_k} - Q(s^{u_i}_k, a^{u_i}_k | \theta^Q))^2 \tag{4.17}$$

$$y^{u_i}_k = R^{u_i}_k + \gamma \; Q'(s^{u_i}_{k+1}, \hat{\mu}(s^{u_i}_{k+1}) | \theta^{Q'}) \tag{4.18}$$

$$\hat{\mu}(s^{u_i}_{k+1}) = \begin{cases} \mu'(s^{u_i}_{k+1} | \theta^{\mu'}) & \textit{with } 1-\epsilon \textit{ probability} \\ \\ \mu^{DINF}(s^{u_i}_{k+1}) & \textit{with } \epsilon \textit{ probability} \end{cases} \tag{4.19}$$

where, $0 \leq \epsilon \leq 1$. $\hat{\mu}(.)$ represents the policy of target actor. Unlike the DDPG algorithm [32], the proposed model makes use of target actions based on DINF flow estimates ($\mu^{DINF}(.)$). ϵ decays as the training progresses.
The learning actor network then gets updated using sampled policy gradient ([32]) as given in Equation 4.20.

$$\nabla_{\theta^\mu} J \approx \frac{1}{\mathcal{B}} \sum_t \nabla_a Q(s, a | \theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s | \theta^\mu)|_{s_t} \tag{4.20}$$

The architecture of the proposed D3S algorithm is illustrated in Figure 4.5. The proposed D3S presented in algorithm 2 highlights how the target actions are generated using the DINF flow estimation algorithm and how these estimates are used to improve the critic network of the DDPG algorithm. This improved critic shifts the gradient of learning actor-network in the correct direction and helps in learning an overall improved policy. As depicted in algorithm 2, the system-related hyperparameters are initialized, such as the

Algorithm 2: D3S Algorithm

```
1  Input: $\mathbb{H}$, $\gamma$, $D$, $\Delta d1, \Delta d2$, soft-update $\eta$, max-time-step $\mathbb{T}$
2  Randomly initialize critic $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
3  Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
4  Initialize Replay Buffer $\mathcal{Z}$
5  for episode=1,2,... do
6      Randomly initialize starting position of the UAVs: $u_i \; \forall i \in U$
7      Receive initial observation state of each agent $s_{t+1}^{u_i} \; \forall i \in U$ (corresponding to
         observed location $c$)
8      while $t \leq \mathbb{T}$ do
9          Select action $a_t^{u_i} = \mu(s_{ct}^{u_i}|\theta^\mu) + \mathcal{N}_t$ according to the current policy $\pi$ and
             exploration noise $\mathcal{N}(0,1)$
10         Execute action $a_t^{u_i}$ and observe reward $R_t^{u_i}$ and observe new state $s_{t+1}^{u_i}$
11         Store transition $(s_t^{u_i}, a_t^{u_i}, R_t^{u_i}, s_{t+1}^{u_i})$ in $\mathcal{Z}$
12         Sample a random mini-batch of $\mathcal{B}$ transitions $(s_k^{u_i}, a_k^{u_i}, R_k^{u_i}, s_{k+1}^{u_i})$ from $\mathcal{Z}$,
             where $k$ denotes the index of $\mathcal{B}$
13         Set $y_k^{u_i} = R_k + \gamma Q'(s_{k+1}^{u_i}, \hat{\mu}(s_{k+1}^{u_i})|\theta^{Q'})$  (as given by Equation 4.18 in
             subsection 4.3.3)
14         The action given by target actor policy is based on $directed - \epsilon$ strategy
             (as given by Equation 4.19 in subsection 4.3.3):
15             $\hat{\mu}(s_{k+1}^{u_i}) = \begin{cases} \mu'(s_{k+1}^{u_i}|\theta^{\mu'}) & with\ 1-\epsilon\ probability \\ \mu^{DINF}(s_{k+1}^{u_i}) & with\ \epsilon\ probability \end{cases}$
16         Update critic by minimizing the loss (as given by Equation 4.17 in
             subsection 4.3.3)
17         Update the actor policy using the sampled policy gradient (as given by
             Equation 4.19 in subsection 4.3.3)
18         Update the target networks:
19             $\theta^{Q'} \leftarrow \eta\theta^Q + (1-\eta)\theta^{Q'}$
20             $\theta^{\mu'} \leftarrow \eta\theta^\mu + (1-\eta)\theta^{\mu'}$
21     end
22 end
```

number of UAVs (n), UAV altitude $\mathbb{H}$, value of γ, number of time-steps $\mathbb{T}$, etc. For each time-step t within an episode, an action $a_t^{u_i}$ is selected for each individual UAV u_i based on its current state $s_t^{u_i}$. Subsequently, the reward received by the UAV and its next state as perceived from the environment are observed. Tuple $(s_t^{u_i}, a_t^{u_i}, R_t^{u_i}, s_{t+1}^{u_i})$ is stored in the replay buffer $\mathcal{Z}$. After the initial experience gathering, the proposed D3S approach is used to learn the control policy for each individual UAV. Random mini-batches of experiences are extracted from the buffer and used to realize the target Q-value $y_k^{u_i}$ based on the state $s_{k+1}^{u_i}$ and the action received from target actor $\hat{\mu}(s_{k+1}^{u_i})$ (see Equation 4.18). Next, the critic loss is calculated using the learning critic Q-value $Q(s_k^{u_i}, a_k^{u_i}|\theta^Q)$ and target Q-value $y_k^{u_i}$ (see Equation 4.17). To learn the optimal policy the learning actor weights are updated

in the direction of the policy gradient, as given in Equation 4.20.
The computation cost of the D3S algorithm as seen from algorithm 2 is $O(E * \mathbb{T})$, where E is the number of episodes and $\mathbb{T}$ is the max_time_step of each episode.
As a Multi-UAV model, the collective information captured by the UAVs is processed at the ground control station to calculate the information gain, system rewards and collision penalty. The proposed D3S algorithm runs individually for each UAV, and hence n different policies are generated by the system. The information processing is done centrally and no UAV-to-UAV communication/information transfer is presumed.

4.4 Experimentation and Results

In this section, the discussion includes the specifications of the environment, details regarding hyperparameters, and the presentation of results.

4.4.1 Training and Test Environment Specifications

As discussed in Chapter 3, section 3.4, the city of Chennai is selected as the training environment. The water level ($\mathbb{L}$) is discretized into 8 flood levels, ranging from 0 to 8 feet. Population density data is obtained using the World Flood Mapping Tool [113]. For the test environment, a grid covering Mumbai city, another coastal city in India [119], is considered. The inundated areas are visualized using the Mapbox API [112].

4.4.2 Hyperparameter Specifications

In all the experiments, the actor network learning rate is set to $1e^{-4}$ and the critic network learning rate to $1e^{-3}$ [19, 120]. 5 quadrotor UAVs are considered that are randomly initialized at distinctive locations in the environment in each episode. The proposed algorithm can be easily extended to more number of UAVs as each UAV learns an individual control policy. The ϵ decay factor is set to 0.9999 having an initial value of ϵ set to 1. The altitude of the UAVs is fixed at 100 meters above the ground. Varying the altitude of the UAV will add another dimension to the action space of the UAVs. Hence, the complexity of learning the control policy for flood area coverage will increase with the change in action dimension. However, the D3S algorithm can still be applied as the proposed strategies of directed exploration and Path scatter are not affected by the varying altitude of the UAVs. The UAVs were assumed to be equipped with identical cameras with half angles θ_1 and θ_2 equal to 30 and 45 degrees respectively. Each experiment is run over 15000 episodes, where each episode is 1000 time-steps long. With each episode, the latitude and longitude data of the environment is slightly altered in order to realize a generalized trend of elevation in the given region. The speed of the water current ($\mathbb{V}_{water}$) is kept constant at 2 m/s throughout the episodes. A digital elevated model (DEM) of the environment is generated using Mapbox.js, and the vector data behind it is queried using the Surface API.
When initializing UAV-related parameters, the following condition is always ensured:

$$\frac{Terrain\ area}{FoV\ area} \geq n \times e_0 \tag{4.21}$$

where, n is the number of UAVs and e_0 corresponds to the UAV energy at t=0 (i.e. maximum energy/fully charged). The value of e_0 should be less than or equal to the number of time-steps in an episode. Equation 4.21 must always hold, or else the multi-UAV system will incur large penalties as UAVs will be more prone to collisions due to insufficient movement space.

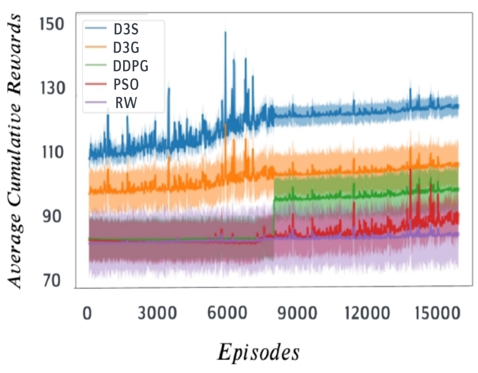
The performance of the proposed D3S algorithm is compared with various other algorithms including coverage algorithms from the literature, namely:

1. Simpler variant, D3G (DINF-DDPG without scatter)
2. Standard DDPG [32] model
3. Random Walk Exploration for Swarm Mapping [121] and
4. Coverage Path Planning of UAVs based on Particle Swarm Optimization (PSO) [122].

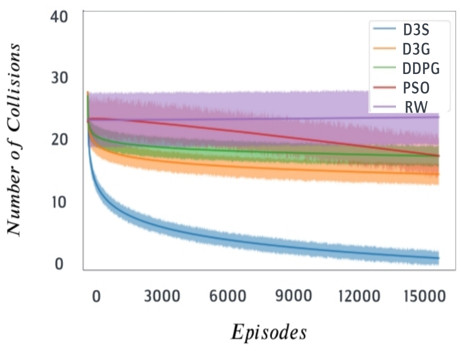
In [121], the authors introduce swarm mapping, where agents initially explore the environment individually using Random Walk (RW), then merge gathered information into a global map. Five RW variants are discussed, namely, Brownian motion, correlated random walk, L'evy walk, L'evy taxis, and ballistic motion. Step-length and turning angle are the two parameters used to generate UAV action sequences. In [122], the authors propose a PSO-based method to optimize yaw-angle and flight altitude for UAVs, addressing area coverage. Each particle represents a configuration i.e., <yaw-angle, flight altitude> that is updated iteratively based on position velocity. The fitness function corresponds to the difference in observed and desired information gain based on the selected actions. Both the techniques, RW and PSO were adapted to generate pitch and yaw angles for comparison with the proposed model. All the experiments are performed on Google Colab having, Intel(R) Xeon(R) CPU, 1xTesla K80 GPU, 2.30GHz CPU frequency, and 12GB RAM.

The results are obtained over 5 different random seeds and the deviation around the mean is also highlighted in the plots. The following performance metrics are used for evaluation:

1. Average cumulative rewards accumulated by the multi-UAV system.
2. Number of collisions observed among the UAVs.
3. Path scatter among the UAVs.
4. UAV path trace where the flight trajectories of the UAVs are analyzed during training.
5. Percentage FoV overlap among the UAVs.
6. Performance on unseen test environment.



(a)



(b)

Figure 4.6: (a) Performance comparison during training based on average cumulative rewards observed by D3S, D3G, DDPG, PSO and RW in a city-scale flooded region. (b) Number of collisions observed by D3S, D3G, DDPG, PSO and RW during training.

4.4.3 Cumulative Rewards Observed during Training

Figure 4.6a depicts the average cumulative rewards observed by various models over the training environment. As seen in Figure 4.6a, RW suffers from the lowest reward accumulation with no noticeable progress in cumulative rewards throughout the episodes. PSO based model also shows little to no progress in terms of reward accumulation until 9000 episodes. However, in later stages, the PSO model is able to achieve relatively larger rewards, that are equivalent to the rewards achieved by DDPG to some degree. This reflects the poor action generation using PSO in the initial episodes and also highlights the fact that PSO easily falls into local optima. The DDPG model observes a significant positive change after 8000 episodes but no further noticeable improvements are observed. DDPG also suffers from poor performance if low rewards are observed in initial episodes which lead to policy updates around poor targets that are far from the global optimum. The proposed algorithms D3S and its lower variant D3G shows significantly improved performances in comparison to other algorithms, especially in initial episodes. This improvement signifies the importance of domain knowledge based learning in realizing improved RL policies. Further, D3S performs the best overall throughout the episodes, highlighting the importance of Path scatter. It helps the model in maintaining a broader coverage over the environment due to which the model is successfully able to visit a larger number of unique locations, hence achieving higher rewards.

In terms of deviation around the mean rewards, D3S results in the lowest deviation, but experiences higher fluctuation in the initial episodes. D3G and DDPG models observe similar deviations followed by PSO, while RW experiences relatively highest deviations around the mean. Such poor performance by the RW model is justified to some degree as random actions are selected at each time-step without any learning. PSO observes relatively higher rewards in later stages but the deviations are still on the higher side. This reflects that higher rewards do not essentially mean stable learning.

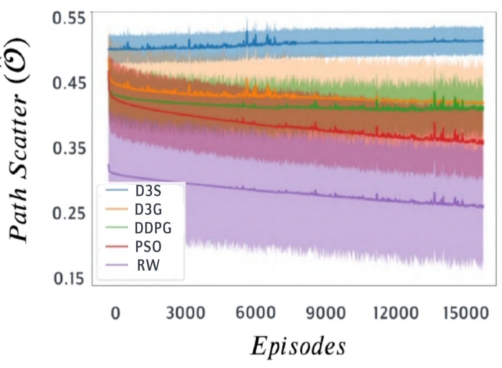


Figure 4.7: Path scatter observed by D3S, D3G, DDPG, PSO and RW during training.

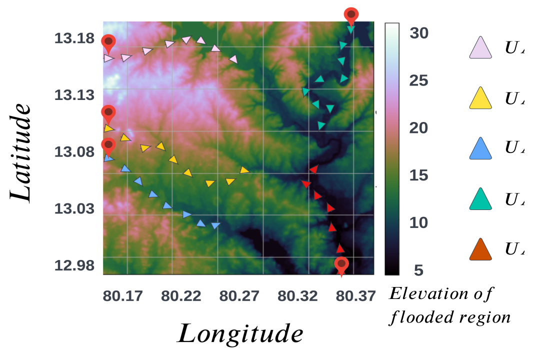
4.4.4 Number of Collisions Observed during Training

In this experiment, second performance metric considered for comparing the models is the number of collisions. As given in Equation 4.5, a collision is recorded whenever two or more UAVs violate the separation threshold D. Figure 4.6b shows the comparison plot highlighting the number of collisions observed by each algorithm during training. As can be seen, the proposed model D3S is able to achieve close to zero collisions in the long run and observes the least number of collisions from the initial episodes itself. There is a logarithmic decrease in the number of collisions in case of D3S. Relatively, a higher number of collisions are observed by D3G even though it performs the second best overall, highlighting the impact of Path scatter in D3S. The Path scatter prompts the UAVs to have broader coverage rather than forming clusters in neighbouring locations which in turn reduces the collision probability.

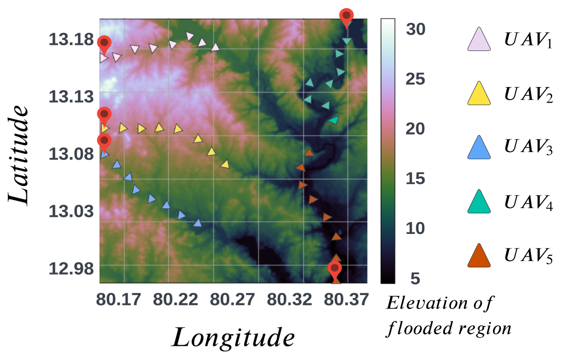
The impact of Path scatter is more noticeable here than in the results observed in the previous section. DDPG shows a similar trend as D3G but observes a higher number of collisions and larger deviation around the mean. The mean number of collisions in case of RW seems to remain almost similar throughout the episodes with the largest deviation as compared to other algorithms. This performance is justified as the UAVs were having random movements with no learning in such a dynamic environment. PSO shows the most varied pattern in the case of collisions where the number of collisions are decreasing linearly, starting as bad as RW and improving to the performance level of DDPG.

4.4.5 Path scatter during Training

Figure 4.7 depicts the observed Path scatter by various algorithms. RW shows the worst performance with the highest deviations across the mean. PSO and DDPG depict a similar trend, but PSO mean values for $\hat{\mathcal{O}}$ are as bad as the lowest values of $\hat{\mathcal{O}}$ of the DDPG algorithm. D3G and D3S show a rather stable and improved Path scatter as the episodes progress. D3S has the least deviation across the mean values of $\hat{\mathcal{O}}$ and records the highest mean value of $\hat{\mathcal{O}}$ overall. Hence, the Path scatter helps the model to disperse over the region in an efficient manner and avoid their clustered formation.



(a)



(b)

Figure 4.8: Path traced by each UAV under (a) D3G (b) D3S policies over the training environment (where different color markers represent the trace and the red pin markers denote the starting positions of the UAVs).

Along with Path scatter observed among the UAVs, the path of each UAV observed over an episode is also highlighted at the end of training. Figure 4.8a and Figure 4.8b depict the path taken by each UAV in the environment when controlled using D3G and D3S respectively. This helps in visualising the effect of Path scatter introduced in D3S. In both scenarios, the starting configuration of the UAVs was kept the same. A noticeable spread can be observed in the trajectories of the UAVs in case of D3S as compared to that of D3G.

4.4.6 Percentage FoV Overlap among UAVs

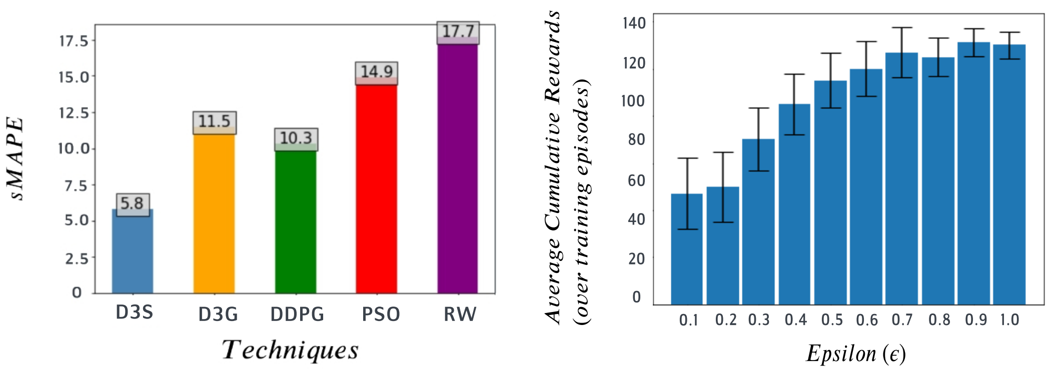
Overlap among the UAVs is also observed by calculating the mean overlap among the FoV pairs during training. An overlap is recorded whenever the distance (d^{v_i,v_j}) between the 2D projected positions of UAVs u_i and u_j goes below the ideal distance D (see Equation 4.5) and is calculated as::

$$overlap^{u_i,u_j} = \begin{cases} \mathcal{F}(u_i,u_j) & if\ d^{u_i,u_j} < D \\ 0 & otherwise \end{cases} \tag{4.22}$$

where $\mathcal{F}(.,.)$ is the intersection area of two FoVs centered at (x^{u_i}, y^{u_i}) and (x^{u_j}, y^{u_j}) [123]. The side lengths of the FoVs are calculated based on the UAV camera half angles θ_1 and θ_2. This overlap is observed as an error recorded over the training episodes using symmetric Mean Absolute Percentage Error ($sMAPE$), calculated as:

$$sMAPE = \begin{cases} \sum_{t=0}^{\mathbb{T}} \sum_{u_i,u_j \in U, u_i \neq u_j} \frac{|overlap^{u_i,u_j}|}{d_t^{u_i,u_j}+D} & if\ d_t^{u_i,u_j} < D \\ 0 & otherwise \end{cases} \tag{4.23}$$

where $\mathbb{T}$ is the total number of time-steps in a single episode. As can be seen in Figure 4.9a, the overlapping error is significantly lower in case of D3S as compared to other algorithms. D3S shows approximately 48% lower overlap than D3G, 43% lower than DDPG, 61% lower than PSO and 67% lower than RW. Even though D3G shows a relatively lower number of



(a) (b)

Figure 4.9: (a) Percentage FoV overlap among UAVs corresponding to D3S, D3G, DDPG, PSO and RW during training. (b) Observed average cumulative rewards corresponding to different initializations of ϵ for D3S.

collisions and higher reward accumulation as compared to DDPG, the percentage overlap is still higher. This highlights the fact that DINF does not prevent the UAVs from clustering together, but it does address the problem of low reward accumulation seen in DDPG by exploiting the domain knowledge.

4.4.7 Effect of epsilon (ϵ)

In this experiment, the focus is on examining how the initial value of ϵ can influence the overall performance of D3S. This analysis facilitates a deeper understanding of the role and impact of the DINF algorithm within the proposed model. Figure 4.9b depicts the average cumulative rewards observed by D3S during training corresponding to different initializations of ϵ. As given by Equation 4.19, a DINF based target action is selected with ϵ probability and a standard DDPG based target action is selected with $1 - \epsilon$ probability. As can be observed from Figure 4.9b, it's better to start with a higher value of ϵ (i.e., close to 1) in the beginning so as to utilize DINF based directed actions. Critic loss calculated using this target helps to boost the policy gradient in the correct direction by improving the learning actor. The decay factor for ϵ is set to 0.9999 which decays the value of ϵ to approximately 0.22 in 15000 episodes.

4.4.8 Policy Generalization in Unseen Test Environment

This experiment is performed to examine the robustness and generalization capabilities of the learnt policies by various algorithms. RW model generates random values for *yaw* and *pitch* within the provided feasible range and PSO based model uses the *yaw* angles and *pitch* values for each UAV that provides the best fitness. Each Deep RL based model uses the same policies learnt during training and no additional learning is performed in this experiment. Figure 4.10a depicts the average accumulated rewards as observed by each model over 100 episodes. Proposed algorithms D3S and D3G performs significantly well as compared to standard DDPG model. Such a noticeable difference in rewards

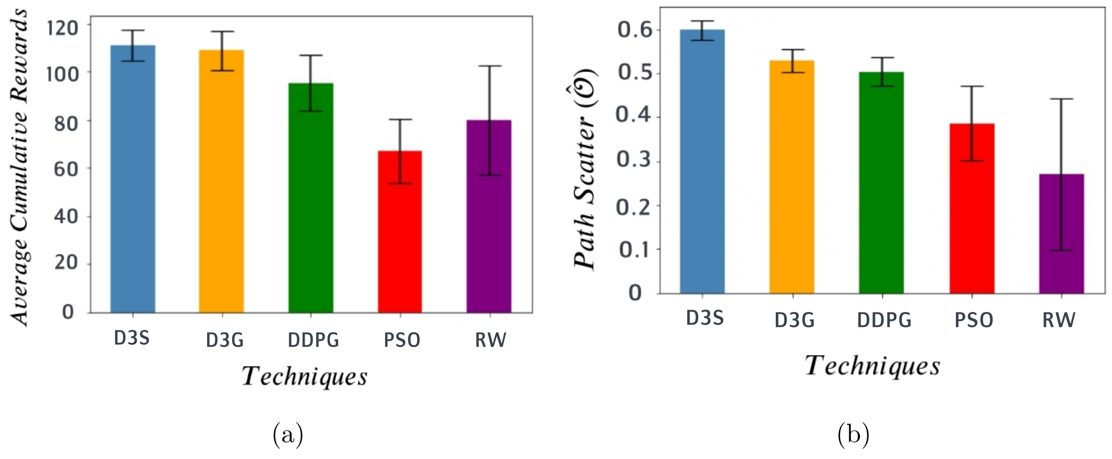


Figure 4.10: (a) Performance comparison based on average cumulative rewards observed by D3S, D3G, DDPG, PSO and RW in an unseen test environment. (b) Path scatter observed over an unseen test environment by D3S, D3G, DDPG, PSO and RW.

highlights the impact of domain knowledge based learning via DINF and in general, the usefulness of domain centric learning to vastly improve the policies learnt by the standard RL algorithms. PSO performing the worst corresponds to the poor generalization ability of the standard PSO algorithm over an unseen environment. The learnt fitness values by PSO corresponding to a very limited number of environment-related parameters may have lead to poor training, especially in case of such highly dynamic environments.

4.4.9 Path Scatter over Unseen Test Environment

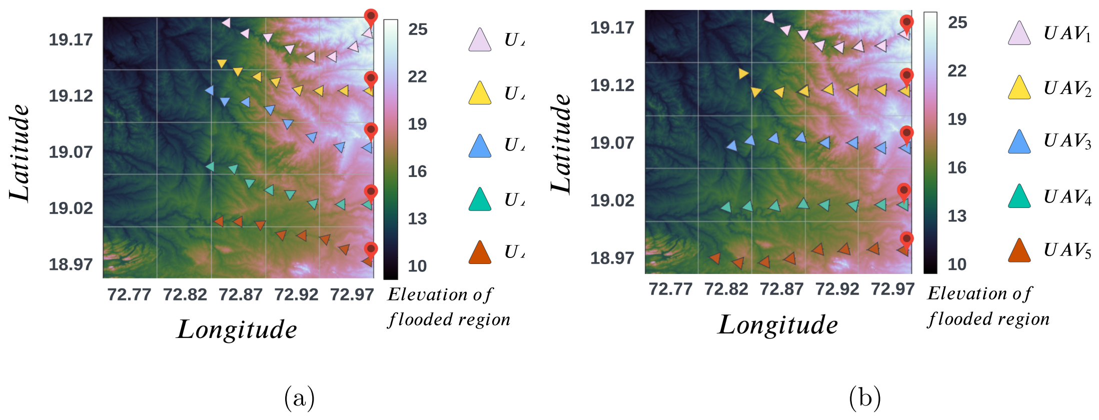


Figure 4.11: Path traced by each UAV under (a) D3G (b) D3S policies over the unseen test environment (where different color markers represent the trace and the red pin markers denote the starting positions of the UAVs).

In this experiment the Path scatter observed by each algorithm is analyzed over the unseen test environment averaged over 100 episodes, as seen in Figure 4.10b. A significant difference can be observed, where, the D3S algorithm records approximately 16% higher Path scatter as compared to D3G. D3S observed around 23% higher Path scatter than the standard DDPG algorithm and about 39% higher Path scatter than PSO. As compared to RW, D3S observed a noticeable 48 (approx.)% higher Path scatter. The UAVs' trajectories over the unseen test environment are also observed corresponding to the two best performing algorithms, i.e., D3S and D3G. Some noticeable differences can be seen from the illustrations given in Figure 4.11a and Figure 4.11b. The initial positions of the UAVs remain the same for both algorithms. The trace trajectories can be seen as stretched out over the terrain in case of D3S as compared to D3G. This highlights the impact of Path scatter as observed visually.

4.5 Chapter Summary

This chapter presents the D3S algorithm, which utilizes the D-infinity water flow estimates to learn the target actions by calculating the exact degree of water flow. Further, D3S prevents UAV clustering using Path scatter. However, the centralized training paradigm followed by D3S (see section 4.2) and D8DQN (see Chapter 3, section 3.4), assume seamless full-duplex communication with a ground control station. This assumption may impose restrictions on efficiently gathering distributed data during disaster scenarios. Issues such as the necessity of a known centralized entity's location in dynamic environments and maintaining bi-directional communication over vast distances might restrict the scale of response operations. Thus, in the next chapter, the discussions are focused on a decentralized training paradigm to learn multi-UAV policies without relying on the ground control station.

5| Autonomous Flood Area Coverage using Decentralized Multi-UAV System

As discussed in the literature review, majority of the prominent work done in the field of multi-agent reinforcement learning (MARL) considers some form of centralized entity to make individual agents learn from global knowledge of the environment [47, 52, 60, 124]. This centralized unit collects information from individual UAVs to expedite training and mitigate the non-stationary effect w.r.t. the multiple agents (UAVs) in the environment. However, deploying such a centralized system to train a multi-UAV policy during disasters might be less effective due to the distributed nature of the data in the environment. In addition, bi-directional communication between UAVs and the central server is required, regardless of their distance, which is usually difficult to achieve in large, dynamic, and stochastic environments such as floods. Decentralized Deep RL approaches have been explored previously in the literature, but limited attention has been given to learning decentralized multi-UAV policies using domain knowledge. In this chapter, a Deep RL algorithm (dec-DQNC8) is proposed to train a decentralized multi-UAV policy for autonomous area coverage in flood environments. Additionally, the D8 algorithm is utilized for directed explorations. Further, communication among UAVs is enabled to exchange their experiences and improve local UAV policies and make them more robust. Coverage maps are also used to make the UAV's aware of the other agents in their vicinity and encourage segregated trajectories for better coverage.
The rest of the chapter is organized as follows. section 5.1 contains the preliminary information highlighting the environment description and the basics of the D8 flow estimation algorithm. section 5.2 presents the system model along with collision avoidance and UAV energy description. section 5.3 presents the proposed methodology, which includes the reward function, action selection, UAV-to-UAV communication, and Coverage Maps. section 5.4 discusses the experiments and results. In section 5.5, a concise summary of the chapter is provided.

5.1 Preliminaries

This section provides a description of the flood environment and the water flow estimation algorithm used to learn exploration strategies for UAVs' policies.

Environment Description

The environment is considered as a 2D terrain divided into $n \times m$ number of cells of equal size (similar to the one considered in Chapter 3, section 3.1). The dimension of a cell is equal to the Field-of-View (FoV) of the UAV. The FoV of a UAV is a rectangular area captured by the UAV's ventral camera and its dimension depends on the UAV's altitude and camera angles. To simulate flood, a 2D water mask is overlayed over the terrain layer. A critical level $\mathbb{Z}_c$ is assigned to each cell based on its water level and population density (for more details, refer to Chapter 3, section 3.1).

Water Flow Estimation

The D8 algorithm [45] is employed that generates the flow estimates based on the estimation of water discharge directions as discussed in Chapter 3, subsection 3.3.2). D8 estimates the cell with the largest water accumulation in the neighbourhood of the cell that is under a UAV's observation. In the considered scenario, the input to the D8 model is the state of the UAV containing the elevation and the water level information of the current and neighbouring cells (8 adjacent neighbours). The cell with the lowest water discharge is usually the one with the highest water accumulation which puts it relatively at a higher risk than others. Once, a neighbouring cell with the lowest water discharge is identified using D8, this information is used to generate an exploration action for the UAV to move in the estimated flow direction (refer to Chapter 3, subsection 3.3.2)). Equation 5.1 denotes the exploration action given by D8 flow estimation technique:

$$a_{c_t}^{D8} = m(L_{c_t}^{u_i}, c^{u_i}) \tag{5.1}$$

where $a_{c_t}^{D8}$ represents the D8 generated action for an individual UAV. The function $m(.,.)$ maps the appropriate action from the feasible action set. $L_{c_t}^{u_i}$ denotes the cell with the lowest relative water discharge in the neighbourhood of c^{u_i}.

5.2 System Model

Having a multi-UAV system of n UAVs $U : \{u_i | i = 1, 2, ..., n\}$, the objective is to capture as many critical regions as possible by performing area coverage under the constraint of limited batteries of the UAVs with minimum overlapping trajectories. UAVs maintain a consistent altitude of $\mathbb{H}$ meters for uniform image resolution. Equipped with ventral cameras each UAV captures a rectangular field-of-view of the surface (refer to Figure 4.1). Each UAV is only able to perceive its local surroundings and the information exchanged by other UAVs through opportunistic communication. They operate without awareness of the global state.

5.2.1 Collision Avoidance

To implement the protocol for collision avoidance among the UAVs, an overlapping constraint ($\mathbb{C}$) is defined to discourage the UAVs from getting into the collision-prone range of each other. The overlapping constraint is given as:

$$\mathbb{C}_t(u_i, u_j) = \begin{cases} 1 & if \quad d_t^{u_i,u_j} \leq \lambda D \ \ \forall \ u_i, u_j \in U \\ 0 & \qquad otherwise \end{cases} \tag{5.2}$$

where, $d_t^{u_i,u_j}$ is the observed Euclidean distance between the 2D projected positions of the UAVs. λD is the threshold distance below which the UAVs are prone to collision. A penalty is imposed on the UAV when its action leads it into a collision prone range with other UAV(s). The collision prone areas are calculated in reference to the coverage maps (CM's) communicated among the UAVs (discussed in subsection 5.3.5). Future locations of the UAVs are estimated based on their CM's, to realize a static path foreseeing a possible collision course with other UAVs (the ones with which it has communicated).

5.2.2 UAV Energy Model

A UAV requires energy for various manoeuvres such as take-off, hovering and flying in addition to the energy required for transmission. Authors in [125] derived an analytical model for propulsion power consumption $\mathbb{W}$ of quadrotor UAVs moving/flying at a speed of $\mathcal{V}$. In [126], authors discuss the energy required to realize the communication transmission between UAVs. Adopting the energy consumption model from [125] and [126], a UAV's energy ψ_t at any given time t can be derived as:

$$\psi_t = \psi_{t-1} - \psi_{\{hover,moving,comm\}} \tag{5.3}$$

$$\psi_{moving} = \mathbb{W}(\mathcal{V}) \times \mathbb{T}_{moving} \tag{5.4}$$

where $\mathbb{T}_{moving}$ is the number of time-steps during which the UAV is in motion.

$$\psi_{hover} = \mathbb{W}(0) \times \mathbb{T}_{hover} \tag{5.5}$$

where $\mathbb{T}_{hover}$ is the number of time-steps during which the UAV hovers.

$$\psi_{comm} = \Omega \times \mathbb{T}_{comm} \tag{5.6}$$

where Ω is the transmission power of the UAV and $\mathbb{T}_{comm}$ is the number of time-steps during which the UAV is transmitting data.

Energy is not part of the reward formulation as the objective is focused on coverage only, however, it plays a crucial role by affecting the episode length/UAV flight time. It limits the UAV actions based on the current energy level of the UAV. For example, if $\psi_t^{u_i} < \psi_{comm}$ the UAV u_i cannot perform communication anymore, similarly for other actions $\{hover, moving\}$.

5.3 Proposed Methodology

In this section, the considered multi-UAV area coverage problem is discussed, along with the action selection strategy, coverage maps and reward function.

5.3.1 MDP Formulation

For a multi-UAV system, the dec-POMDP is given by a tuple $< n, S, \{A^{u_i}\}, P^T, \{O^{u_i}\}, \{R^{u_i}\}, P^O, \Gamma >$ where n is the number of UAVs and S denotes a finite set of hidden states. $A^{u_i} = \{N, S, E, W, NE, NW, SE, SW, hover, hover + comm\}$ (comm is the abbreviation for communication) is the finite action set. P^T is the state transition probability that provide the distribution $P^T(s_{t+1}|s_t, a_t)$ of transitioning to the next state s_{t+1} given the current state s_t and joint action $a_t = \{a^{u_1}, a^{u_2}, ..., a^{u_n}\} \in \mathbb{A}$. $\mathbb{A}$ is the joint action space of all the UAVs. $O^{u_i} = \{o_1^{u_i}, o_2^{u_i}, ...\}$ is the finite observation set for each UAV u_i, where a single observation is represented as:

$$o_t^{u_i} : \{\mathbb{P}^{u_i}_{c+k_1,c+k_2}, e^{u_i}_{c+k_1,c+k_2}, h_c| \ \forall \ k_1, k_2 \in \{-1, 0, 1\}\}$$

where $\mathbb{P}_c^{u_i}$ corresponds to the human population density level of cell c and $e_c^{u_i}$ corresponds to the terrain elevation. h_c is the water level at the sensed location c. The $k1, k2 \in \{-1, 0, 1\}$ set represents the elevations and human population density levels of the neighbouring 8 cells of c. P^O is the observation probability given the conditional probability $P^O(o_{t+1}|s_{t+1}, a_t)$. Set R^{u_i} defines the accumulated rewards earned by each UAV $\forall \ i \in U$ present in the system. At each time-step t, the UAV takes an action a^{u_i} based on its local observation history $o_{[1:t]}^{u_i}$ and receives the next observation $o_{t+1}^{u_i}$ and rewards $R_t^{u_i}$ from the environment. As the complete state of the environment is unknown to the UAV, the observation history is useful in realizing a local policy π^{u_i}. So, π^{u_i} can be defined as the mapping from local histories to agent-specific actions and the joint policy $\Pi : \{\pi^{u_1}, \pi^{u_2}, \pi^{u_i}...\}$ is the set of local policies corresponding to each UAV $u_i \in U$.

5.3.2 The Value and Reward Functions

The objective is to find the optimal policy π that achieves the maximum cumulative rewards in the long run. The state-action value function $Q^\pi(s_t, a_t)$ under the policy π defines the long-term desirability of an action in a particular state, given as

$$Q^\pi(s_t, a_t) = \mathbb{E}_{a_t \sim \pi} \left[\sum_{l=0}^{\Gamma} \gamma^l R_{t+l+1} | s_t, a_t \right] \tag{5.7}$$

where $0 \leq \gamma < 1$ is the discount factor (used to maintain finite sum over the infinite horizon). In the considered scenario, as the UAVs are unaware of the global state, the Q-value is defined in terms of the global expected utility as the sum of the local utilities of each UAV.

$$Q(O_t^N, a_t) = \sum_{u_i, u_j \in U} Q^{u_i}(o_{[1:t]}^{u_i}, a_t^{u_i}, a_t^{u_j}) \tag{5.8}$$

where O_t^N is the joint observation set at time t, a_t is the joint action. $o_{[1:t]}^{u_i}$ is the local observation history of UAV u_i up until time t. $a_t^{u_i}$ corresponds to the local action of i^{th} UAV at time t and $a_t^{u_j}$ denotes the action of UAV $(u_j \in U)$ that interacted with the u_i at time t (a UAV can communicate with only a single agent at any given time).

The formulation of the proposed reward function is based on the information gained by the UAV from the environment. The other parameter is whether the UAV is communicating or not and whether it's on a collision course with another UAV or not. The information gain is a quantitative measure that defines the importance of the cell c captured by the UAV. Higher the risk level of a cell, higher is the information gain. Hence the information gain ($I_c^{u_i}$) from a cell c as observed by a UAV u_i is given as:

$$I_c^{u_i} = \frac{\mathbb{Z}_c}{max(\mathbb{Z})} \tag{5.9}$$

where $\mathbb{Z}_c$ is the risk level of cell c. The reward $R_t^{u_i}$ corresponding to an individual UAV u_i at time t is calculated as:

$$\begin{aligned} R_t^{u_i}(s_t^{u_i}, a_t^{u_i}, s_{t+1}^{u_i}) = I_c^{u_i} + C_c^{u_i} - \alpha_t(u_i, u_j).\mathbb{C}_t(u_i, u_j) \\ \forall \;\; u_j \in N, j \neq i \end{aligned} \tag{5.10}$$

where $C_c^{u_i}$ is a positive scalar incentive given to u_i provided that it is successfully able to build a communication link with another UAV at time t. $\alpha_t(.,.)$ is a function that outputs a scalar penalty when a UAV u_i's action leads it into a collision-prone range. The penalty associated with the function $\alpha(.,.)$ exceeds $C_c^{u_i}$ due to the emphasis on collision prevention rather than prioritizing communication. Function $\mathbb{C}(.,.)$ denotes whether two UAVs are within the collision range or not (the communication range is greater than the collision range).

5.3.3 Action Selection

In the standard DQN model [13], the action selection is based on the ϵ-greedy strategy. As initially, the environment is completely unknown, random actions are performed to explore the value of different actions from various states. In later stages of training, the agent exploits the already gathered information to perform the given task in an optimal manner. However, random exploration can lead to a sub-optimal policy as it is the least efficient exploration method when it comes to limited energy models [127]. In the proposed approach nicknamed dec-DQNC8, a better exploration strategy is adopted based on the D8 flow estimation algorithm [45]. The employed action selection strategy (for more details refer to Chapter 3, subsection 3.3.3) is given as:

$$a_t^{u_i} = \begin{cases} \underset{a'}{argmax}\; Q(o_{[1:t]}^{u_i}, a') & 1 - (\epsilon_1 + \epsilon_2)\; probability \\ a_{ct}^{D8} & \epsilon_2\; probability \\ random \;\; action & \epsilon_1\; probability \end{cases} \tag{5.11}$$

where, $0 \leq \epsilon_1, \epsilon_2 \leq 0.5$ u_i denotes the i^{th} UAV and a' denotes the action given by the target network. a_{ct}^{D8} denotes the action based on D8 flow estimation algorithm.

Algorithm 3: dec-DQNC8 Algorithm

```
1  Input: $n$, $\Omega$, $f_c$, $\mathcal{V}$, $\gamma$
2  Initialize action value function $Q^{u_i}$ with random weights $\theta^{u_i}$ for each UAV $i \in U$
3  Initialize target action value function $\hat{Q}^{u_i}$ with weights $\theta^{-} = \theta^{u_i}$ for each UAV
     $i \in U$
4  for $UAV = u_i, u_j, ..., u_n$ do
5      for episode=1,2,... do
6          Initial observation $o^{u_i}$ and coverage map $\hat{m^{u_i}}$
7          while $t \leq max\_time\_step$ and $\psi_t > \psi_{\{hover,moving,comm\}}$ do
8              Select action $a_t^{u_i}$ as given in Equation 5.11
9              Decay UAV's energy as given in Equation 5.3
10             UAV $u_i$ makes the next observation $o_{t+1}^{u_i}$ and receives reward $R_t^{u_i}$ from
                 the environment.
11             Store transition $< o_t^{u_i}, a_t^{u_i}, R_t^{u_i}, o_{t+1}^{u_i} >$ in replay buffer $\mathcal{Z}^{u_i}$
12             if $a_t^{u_i} == hover + comm$ then
13                 $\mathcal{Z}^{u_i}.append(\mathcal{Z}^{u_j})$    // Considering $u_j$ and $u_i$ communicated at time
                     $t$
14                 Update the input state as given in Equation 5.16
15             end
16             Sample a random mini-batch of $\mathcal{B}$ transitions $(o_k^{u_i'}, a_k^{u_i'}, R_k^{u_i'}, o_{k+1}^{u_i'})$ from
                 $\mathcal{Z}^{u_i}$, where $k$ denotes the index of $\mathcal{B}$ and $u_i'$ denotes the experience of
                 $u_i$ or the experience of any other UAV that communicated with $u_i$.
17             Calculate target Q value:  $y_k^{u_i} = R_k^{u_i'} + \gamma \max_{a'} \hat{Q}(o_k^{u_i'}, a'; \theta^{-})$
18             Perform gradient decent step on $(y_k^{u_i} - Q(o_k^{u_i'}, a_k^{u_i'}; \theta))$ w.r.t. network
                 parameter $\theta$
19             In every $C$ steps, reset $\hat{Q}^{u_i} = Q^{u_i}$
20         end
21     end
22 end
```

Figure 5.1: Illustrating proposed architecture of dec-DQNC8 algorithm.

$o_{[1:t]}^{u_i}$ is the local observation history of UAV u_i up until time t. UAV's action $(a_t^{u_i})$ is subjected to its available energy (refer subsection 5.2.2). The proposed dec-DQNC8 is depicted in algorithm 3. As can be noted from line number 1-3,the system related hyperparameters and the network weights are randomly initialized. Next, within each episode, the observations of each UAV w.r.t. the captured image of the environment is recorded and marked in individual coverage maps (refer to line number 4-6). Then an action is performed by the UAV(s) based on its current policy, as depicted in line number 8. Later, when enough experience is gathered by local movements and experience gained from communication, the policy network is updated by minimizing the loss and shifting the gradient in the correct direction (refer to line number 11-18). The model's architecture is illustrated in Figure 5.1.

5.3.4 UAV-to-UAV Communication

A UAV u_i can transmit the replay buffer $\mathcal{Z}^{u_i}$ and coverage map $\hat{m}^{u_i}$ in form of packets to another UAV u_j if the UAVs are within a transmission range $\mathbb{D}$ of one another [128]. It is assumed that the UAVs are equipped with radio transmitters and work on 2.4 GHz to 5.8 GHz frequency. This range is calculated as:

$$\mathbb{D} = 10^{-[10 \times log_{10}(\zeta^{u_i,u_j}\Upsilon/\Omega)+28+20\times log_{10}(f_c)]/22} \tag{5.12}$$

where ζ^{u_i,u_j} is the threshold SNR. Ω is the transmission power of the UAV and Υ denotes the thermal noise described as the Gaussian white noise [129]. f_c is the carrier frequency of the communication signal [130]. A packet is successfully received if the mean SNR $\hat{\zeta}^{u_i,u_j}$ is greater than the threshold SNR ζ^{u_i,u_j}.

$$\hat{\zeta}^{u_i,u_j} = \frac{\Omega}{\Upsilon} \times 10^{-\frac{28+22\times log_{10}(h^{u_i,u_j})+20\times log_{10}(f_c)}{10}} \tag{5.13}$$

where h^{u_i,u_j} is the fading coefficient of communication link between UAV u_i and u_j based on Nakagami-m distributions [131]. Whenever two UAVs (u_i, u_j) are within a distance $\mathbb{D}$ of each other, both the UAVs perform *hover* + *comm* action to transmit the replay buffer information and their coverage maps to each other (conditioned on whether enough energy is left for transmission or not). After transmission completes, UAV u_i's replay buffer becomes equal to $\mathcal{Z}^{u_i}.append(\mathcal{Z}^{u_j})$ (and similarly for UAV u_j). This increase in information helps the dec-DQNC8 model to converge early and the resultant policy is more robust as it is learnt from a more diverse and distributed form of data. As the replay buffer data is shared among UAVs over communication, the individual policies learnt by the UAVs could be overlapping to some extent. However, it is highly unlikely for two or more UAVs to conclude training with very similar weights. This is primarily due to the differences in their initial weights, that are randomly initialized, and the rarity of these UAVs having precisely the same data to train from in each episode.

5.3.5 Coverage Maps

In this section, the use of Coverage Map (CM) is discussed that contains the local trace of the UAV based on its local observation history. This map is also shared among the UAVs when they communicate. A coverage map contains the information corresponding to the locations that a UAV has visited during its flight and also the recent time-step at which that particular cell was observed. The data contained in the map is used to update the information gain, effectively altering the rewards that a UAV can accumulate from a cell (this is applicable to the time steps after the communication has occurred). The updated information gain is calculated as:

$$I_c^{u_i} = \frac{\mathbb{Z}_c}{max(\mathbb{Z})} \cdot \frac{t_c^{u_i} - t_c^{u_j}}{t_c^{u_i}} \tag{5.14}$$

In reference to the UAV u_i currently observing cell c, the updated information gain $I_c^{u_i}$

is calculated corresponding to $t_c^{u_i}$ (the current time-step) and $t_c^{u_j}$ ($u_j \in U$, the UAVs that have communicated and transferred their coverage maps to u_i up until time t). $t_c^{u_j}$ denotes the last time-step at which the cell c was observed, as recorded in the CM. The idea is to diminish the rewards of a UAV for the cells that have been previously visited by the other UAVs. This helps to learn a local policy by a UAVs to emphasizes on visiting more unobserved cells rather than revisiting the observed ones.
Further, the future locations of a UAV are estimated (in reference to u_j) based to its map $\hat{m}_t^{u_j}$, calculated as:

$$C_{t+1}^{u_j} = \underset{c_k^{\eta_j}}{argmax}(M_t^{u_j} \mid \hat{m}_t^{u_j}) \tag{5.15}$$

where, $C_{t+1}^{u_j}$ represents the location of UAV u_j at time-step $t+1$. $\underset{c_k^{\eta_j}}{argmax}(.|.)$ depicts the cell with the highest water accumulation (usually the lowest elevation cell) in the neighbourhood ($c_k^{u_j}$) of u_j. To make sure that the communicated trace information is available to the UAVs during training (at all times), the coverage map information is integrated as part of the model's state input. A coverage map also has a huge impact when the model's performance is evaluated in a test environment (discussed in subsection 5.4.3 and subsection 5.4.4).
The updated information gain as given in Equation 5.14 only corrects the future rewards considering the cells that are being revisited. However, the already observed rewards also needs to be updated that are in the replay buffer. Let's say at time t UAV u_i communicated with u_j (a UAV can communicate with only a single UAV at a time), the replay buffer of u_i becomes equal to $\mathcal{Z}^{u_i}.append(\mathcal{Z}^{u_j})$. Based on the number of overlapping cells between u_i and u_j up until time t as observed from the traces in $\hat{m}^{u_i}$ and $\hat{m}^{u_j}$, the rewards in the replay buffer are updated in reference to the Equation 5.10 and Equation 5.14. For a UAV u_i the updated state contains the local state information, its coverage map and the coverage map of the UAVs with which u_i communicated.

$$s_{input}^{u_i} = \{o^{u_i}, \{\hat{m}^{u_i}, \hat{m}^{u_j}\}\} \ j \in U \ ; \ j \neq i \tag{5.16}$$

5.4 Experimentation and Results

In this section, the performance of the proposed model dec-DQNC8 is evaluated against the simpler variant dec-DQN8 and two other state-of-the-art multi-UAV coverage techniques from the literature, namely, dec-DQN [28] and PSO [132]. Communication based reward updates are not implemented for dec-DQN8, and the utilization of coverage maps is also omitted. In [28], authors employ a DQN-based approach to learn multi-UAV controls for flood monitoring. The proposed technique is said to be decentralized but no communication method is introduced or employed. When comparing with the proposed model, the algorithm presented in [28] is regarded as a decentralized DQN without communication, where UAVs are only aware of their local state. In [132], the authors

proposed a distributed PSO model to perform exploration of a disaster area using UAVs. There is no central node considered in this distributed approach, but the UAVs are able to share their local information with other agents that are in its vicinity.
To evaluate dec-DQNC8, a team of 7 UAVs (quadrotors) is considered during experimentation. The altitude of the UAVs is fixed at 100 meters above sea level. Considering a standard UAV camera, the half angles are assumed to be 30$\circ$ and 45$\circ$. To create the training environment, the coastal region of Chennai city is considered, similar to the one considered in Chapter 3, section 3.4. A different test environment is simulated to evaluate the learnt policies of the multi-UAV system. The water level information and population density information is encoded in a similar manner as discussed in Chapter 3, section 3.4. The collision range for UAVs is set to 10 meters, under which UAVs are bound to collide. Implementation is done on Google Colab having, Intel(R) Xeon(R) CPU, 1xTesla K80 GPU, 2.30GHz CPU frequency, and 12GB RAM. The following performance metrics are used to evaluate the proposed model:

- Average cumulative rewards observed by the multi-UAV system over the training and test environments.
- Average joint coverage observed during training and testing.
- Multi-UAV path trace observed over the training and test environments.

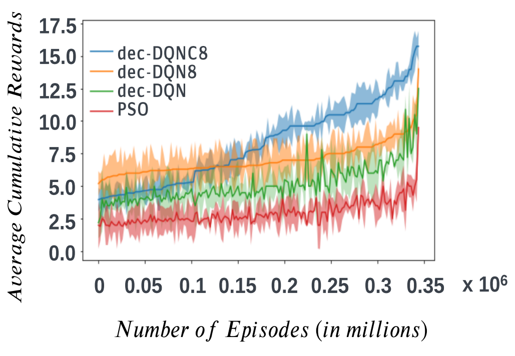
Results are observed over 5 different random seeds and the deviation around the mean is highlighted in the plots.

5.4.1 Average Cumulative Rewards Observed During Training

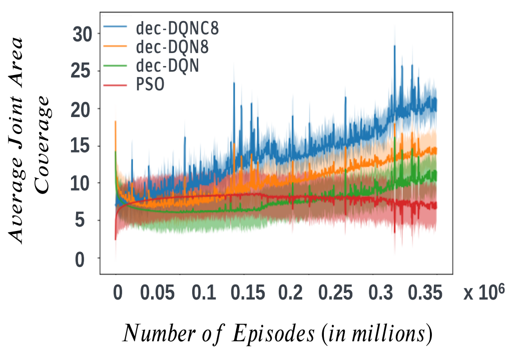
Experiments are performed over the course of 0.35 million episodes to observe the difference in cumulative rewards more vividly (if present). Each episode is of 1000 time steps. Figure 5.2a depicts the average cumulative rewards observed by various models during training. As observed, all the RL-based methods perform similarly in the initial episodes with dec-DQN8 performing the best. But, as the number of episodes progresses dec-DQNC8 outperforms the others to achieve the highest overall rewards. Such behaviour of dec-DQNC8 is justified as the rewards are updated within episodes whenever the UAV communicates, leading to a decline in average rewards. However, as soon as the coverage map becomes good enough after communication, UAVs are able to spread over the environment in a much better way, covering a significantly larger area. PSO performs the worst highlighting the pitfall of PSO as it usually gets stuck in local optima, especially when training is done using a limited number of environment parameters.

5.4.2 Average Joint Coverage Observed During Training Along with the Multi-UAV Path Trace

In this experiment, the average joint coverage of the environment is observed during training. The coverage is defined as the number of unique cells captured by the UAVs



(a)



(b)

Figure 5.2: Performance comparison between dec-DQNC8, dec-DQN8, dec-DQN and PSO based on (a) average cumulative rewards and (b) average joint coverage during training.

in an episode. Higher coverage provides a better chance of capturing a relatively larger number of critical regions. Figure 5.2b depicts the average coverage by the multi-UAV system during training. As observed, up until the 0.15 million episodes there is no significant change in coverage to separate the models, but after 0.2 million (approx.) episodes, dec-DQN shows a noticeable improvement in the joint coverage as it sees a linear rise, outperforming PSO. A noticeable gap in joint coverage can be observed between dec-DQNC8 and dec-DQN8 after 0.25 million episodes and this gap seems to increase with the increase in the number of episodes. This signifies the impact of the coverage map used by dec-DQNC8 in achieving a better spread over the environment.

To further analyze the impact of updating rewards and sharing coverage maps in learning a multi-UAV policy, the paths of the UAVs during training for both dec-DQNC8 and dec-DQN8 are traced. dec-DQN8 adopts the D8 flow estimation strategy for better exploration and shares the experience replay buffer across UAVs during communication. But, it does not update the rewards of the replay buffer on communication and also does not use the coverage map for policy learning. Figure 5.3a and Figure 5.3b illustrates the multi-UAV path observed during a single episode at the end of the training by both dec-DQNC8 and dec-DQN8. As observed, the updation in rewards and the use of CM helps in maintaining better coverage over the environment in the long run. Rather than forcing a constraint on the multi-UAV system to maintain inter-UAV separation, here the system learns naturally about the effects of clustering with other UAVs.

5.4.3 Average Cumulative Rewards Observed During Testing

To analyze the generalizability and robustness of the learnt policy using dec-DQNC8, the performance of the model is observed in a test environment over 100 episodes. During testing, the learnt policies are not updated, but the UAVs are able to share their coverage maps with each other during communication (in the case of dec-DQNC8). The test environment is simulated using the real-world elevation data of the Barpeta district of Assam which is one of the most prone regions to floods. Figure 5.4a depicts the

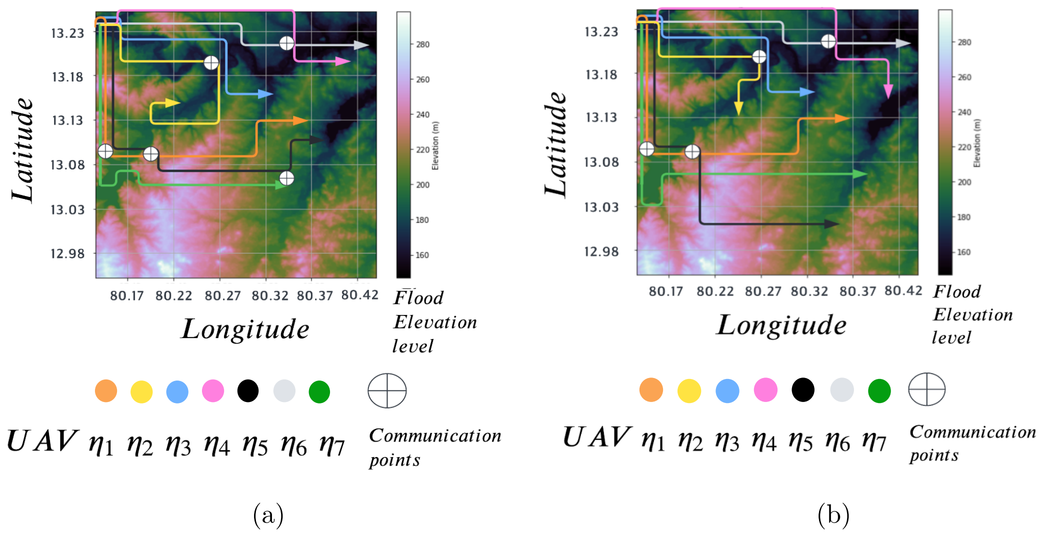


Figure 5.3: The multi-UAV path trace observed during training with (a) No updation in rewards and (b) Updating rewards within episodes in the experience replay buffer.

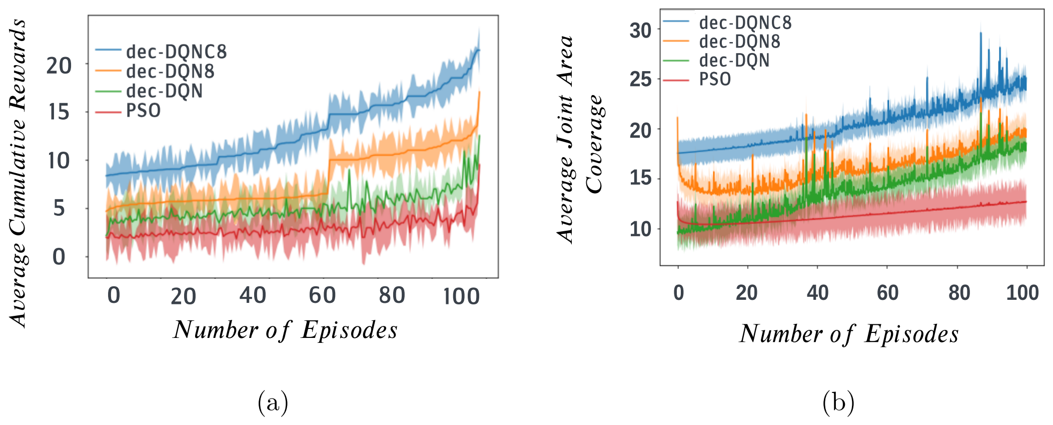


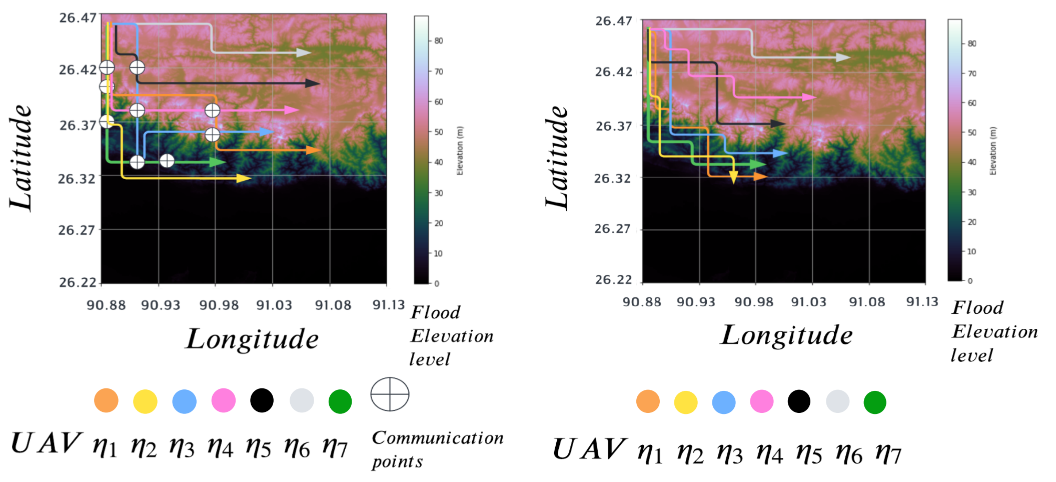
Figure 5.4: Performance comparison between dec-DQNC8, dec-DQN8, dec-DQN and PSO based on (a) average cumulative rewards and (b) average joint coverage during testing.

average cumulative rewards observed by various algorithms during testing. As observed, dec-DQNC8 has the best performance from the initial episode itself. This highlights the fact that the proposed model is able to learn a robust multi-UAV policy. dec-DQN8 also achieves significant rewards during testing. This justifies that the D8-based models are able to learn a better policy and achieve relatively higher rewards even in a short duration of time as compared to dec-DQN. PSO and dec-DQN have similar performances, with PSO performing the worst in later episodes.

5.4.4 Average Joint Coverage Observed During Testing Along with the Multi-UAV Path Trace

In this experiment, the average joint coverage is observed in the test environment. This helps in highlighting the difference in the learnt policies and to observe whether the multi-UAV system is able to maintain a considerable spread over the environment or not when the policies are fixed. As observed from Figure 5.4b, the proposed model dec-DQNC8 sees better coverage as compared to other algorithms. This highlights the impact of using the coverage map to learn the policies by including the map as part of the input state. dec-DQN8 sees better coverage as compared to dec-DQN up until 60^{th} episode. PSO sees the worst performance, highlighting the difficulty in learning suitable control sequences in a highly dynamic environment.

To analyze the significance of communication during testing, two scenarios are considered, one where the UAVs can communicate with each other and the other where the communication is restricted. Figure 5.5a and Figure 5.5b represent these two scenarios where the multi-UAV path trace is observed during testing. As can be seen, if the UAVs are allowed to communicate they are able to spread significantly better over the test environment just by sharing their coverage maps. This results in a higher probability of covering a larger number of critical regions.

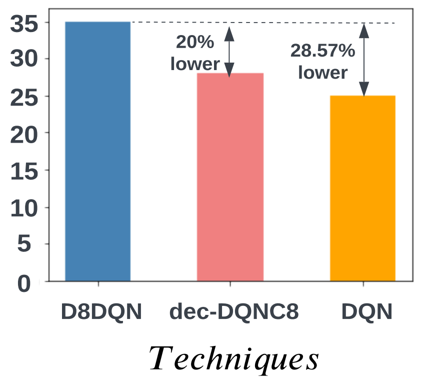


(a) (b)

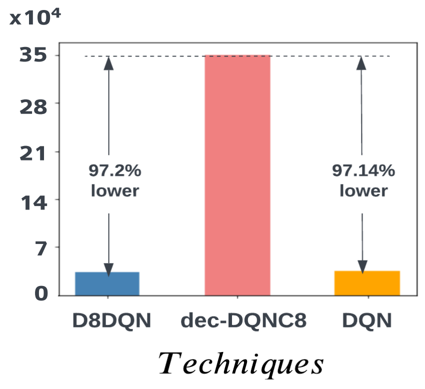
Figure 5.5: The multi-UAV path trace observed during testing with (a) allowable inter-UAV communication and (b) no communication.

5.4.5 Comparison with Centrally Trained Policies

For the performance gap between centralized and decentralised policies, the proposed algorithm dec-DQNC8 is compared with D8DQN and DQN [44] over the average joint area coverage and the number of episodes required to converge during training, as depicted in Figure 5.6a and Figure 5.6b respectively. All three models are trained and observed under identical environmental conditions. As depicted in Figure 5.6a, the centralized approach (D8DQN) achieves 20 % larger coverage as compared to dec-DQNC8. Such an outcome can be intuitively analyzed since in the case of D8DQN all the UAVs are jointly regulated by a central system whereas, in the case of dec-DQNC8, the UAVs are dependent on experience sharing via opportunistic communication. However, the effect of utilizing domain knowledge using D8 has a significant impact on the models, since dec-DQNC8 achieves 8.5 % larger area coverage than the standard DQN model (i.e., a centralized



(a)



(b)

Figure 5.6: Centralized (D8DQN) vs Decentralized (dec-DQNC8) system comparison based on (a) average joint coverage and (b) number of episodes to converge during training.

approach). In terms of convergence, all three approaches do converge but at different rates. D8DQN and DQN only took about 2.8 % (approx.) of the number of episodes to converge as compared to dec-DQNC8. Such a large gap signifies the difficulty in training a decentralized model as compared to a centralized one. But knowing that a decentralized based approach also converges is a step towards realizing real-world models.

5.5 Chapter Summary

Decentralized Deep RL models used in UAV training face several challenges such as slow convergence, limited local data, high communication overheads, inefficient exploration, computational constraints, and lack of centralized coordination, leading to sub-optimal policies. Nevertheless, successful training of a decentralized multi-UAV system using the proposed dec-DQNC8 algorithm was achieved despite these limitations. This technique proves crucial in scenarios where information is widely distributed across the environment and establishing communication with a ground control unit becomes unfeasible. The upcoming chapter extends the application scope to discuss multi-UAV assisted real-time path planning for rescue teams aiming to reach these critical locations during floods.

6| Real-Time Serviceable Path Planning during Floods

After identifying the critical regions in the flood-struck area, the problem of path planning is addressed to assist waterborne evacuation vehicles (WBVs) to reach these critical locations using UAVs. Autonomous navigation and formation control of multi-UAV systems poses a significant challenge for the robotic systems that operate in unknown, dynamic and stochastic environments. In this chapter, the objective is to deploy multiple UAVs in an interleaved formation to identify serviceable paths in an unknown flooded region. The objective is divided into two tasks: firstly, capturing environment related information in connected regions using UAVs and latter, utilizing this information to realize serviceable paths to reach critical locations. UAVs need to capture connected locations in their joint Field-of-View (FoV) to avoid coverage gaps so as to ensure that a serviceable path exists. Multi-agent Deep Deterministic Policy Gradient (MADDPG) algorithm is employed for the multi-UAV system to achieve autonomous motion and interleaved formation. The UAVs are tasked to capture the obstacle-related data and identify shallow water regions for unrestricted motion of the WBV(s). After gathering this information, MEA* (minimum expansion A*) algorithm is used for path planning. MEA* address the node expansion issue with the standard A* algorithm, by pruning the unserviceable nodes/locations based on the captured information, hence expediting the path planning process.

The rest of the chapter is organized as follows. section 6.1 contains the environment description. section 6.2 describes the multi-UAV model along with objective function. section 6.3 presents the proposed Minimal Expansion A* (MEA*) approach with a node expansion strategy, that utilizes information sensed by the UAVs. section 6.4 discusses the experiments and results. Finally, in section 6.5, a concise summary of the chapter is provided.

6.1 Environment Description

The flood environment consists of an underlying surface seen as a grid with $m_1 \times m_2$ cells with identical side dimensions. The surface is encoded with real-world road-transit network information along with land use data (such as buildings, water bodies, etc.) gathered using Mapbox Streets [133]. Further, to simulate flood-like dynamics, a 2D water mask is defined over the environment. The water level at different cells varies within a fixed range to imitate flooding motion. The water level depth and the detection of small underwater objects/infrastructure are sensed through bathymetry [105]. By measuring shallow water depths, captured locations (sensed by a UAV) are categorized as serviceable or not.

Serviceable Path: The aim is to identify serviceable paths for WBV to reach critical

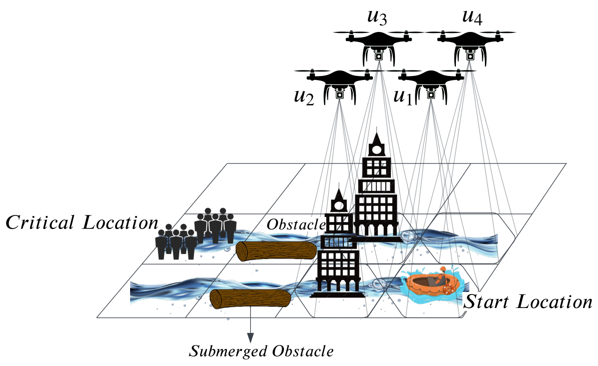


Figure 6.1: End-to-end UAV coverage to identify a serviceable path for WBV to reach a critical location.

location(s). A location c is said to be serviceable Λ^c if the underlying surface is clear of any underwater objects/infrastructure and the water is not shallow for possible movement of WBV.

$$\Lambda^c = \begin{cases} 1(serviceable) & if\ h_c \geq \mathcal{W}\ and\ O^c == False \\ 0(unserviceable) & otherwise \end{cases} \tag{6.1}$$

where O^c denotes whether the location c has an underlying or surrounding obstacle/object(s) and h_c is the water level/depth at location c. $\mathcal{W}$ denotes the threshold level below which the water at cell c is said to be shallow (unsafe for WBV movement, as the boat could get stuck).

6.2 System Model

In the system model, there is a team of n UAVs (U): $\{u_1, u_2, ..., u_n\}$ that are tasked to perform area coverage for serviceable path planning for a waterborne vehicle(s) (WBV) by sensing the region between the start location (i.e., location of the vehicle) and the location of critical areas. The start and critical locations are always known. The objective is to find connected locations (interlinked cells of the environment grid) so as to make the WBV reach the critical location(s). Figure 6.1 describes the considered scenario where a team of 4 UAVs (u_1, u_2, u_3, u_4) are sensing the cells ahead of the WBV in the direction of the closest critical location to identify a serviceable path for the WBV to move. The WBV is perceived in the joint FoV at all times to identify the start location. In this sense, the start location keeps on changing due to the movement of the WBV. Each UAV is equipped with ventral cameras to capture the underlying surface in their field of view (FoV). The FoV of each UAV forms a 3D pyramid having a hexagonal field of view as shown in Figure 6.2. The joint FoV of the UAVs is given by the set $F = \{v^{u_1}, v^{u_2}, ..., v^{u_n}\}$, where, $v_i : \{x_i, y_i\}$ is the FoV of the i^{th} UAV centered at (x_i, y_i).

The overall objective of the system for maintaining formation amongst UAVs and serviceable path planning is given as:

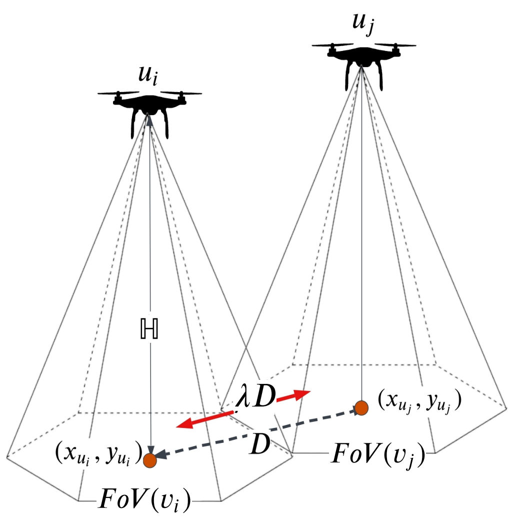


Figure 6.2: UAVs u_i, u_j with a joint hexagonal field of view (FoV) and ideal overlap.

$$\max \sum_{t=0}^{\infty} \left(\sum_{u_i \in U} \mathbb{T}_t(u_i, \Psi)\Lambda^c + \sum_{v_i, v_j \in F} \mathbb{O}_t(v_i, v_j) + \varphi \Lambda^c \right) \tag{6.2}$$

where, t denotes time. $\mathbb{T}(.,.)$ is the trajectory alignment reward w.r.t. the guided path (Ψ) given by the shortest Manhattan distance from start to critical location. $\mathbb{O}(.,.)$ represents the FoV overlap incentive so that the UAVs learn to maintain end-to-end coverage with no gaps. φ is a scalar incentive given to the multi-UAV model if the sensed cells provide a serviceable path.

The UAVs have limited flight time due to finite energy that depletes at a constant rate (similar to the energy model adopted in Chapter 3, section 3.2), given as:

$$\psi_{t+1}^{u_i} = \psi_t^{u_i} - \Delta\psi \tag{6.3}$$

where $\psi_t^{u_i}$ is the energy of u_i at time t and $\Delta\psi$ represents the energy depletion per unit of time. The UAVs are encouraged to follow the guided path only when it is obstacle free (as constrained by Λ^c). If an obstruction is captured by a UAV, the path is no longer deemed to be serviceable and the UAVs help in adjusting the WBV trajectory (i.e., the guided path) by sensing neighbouring regions (see Figure 6.3).

6.3 Proposed Methodology

This section begins with the presentation of the MDP, followed by multi-UAV control policy and path planning algorithm.

6.3.1 MDP Formulation

The multi-UAV coverage task is formulated as an MDP tuple $< S, A, P_{ss'}, R, s_0, s_{start}, s_{critical} >$, where S denotes the joint state of the UAVs

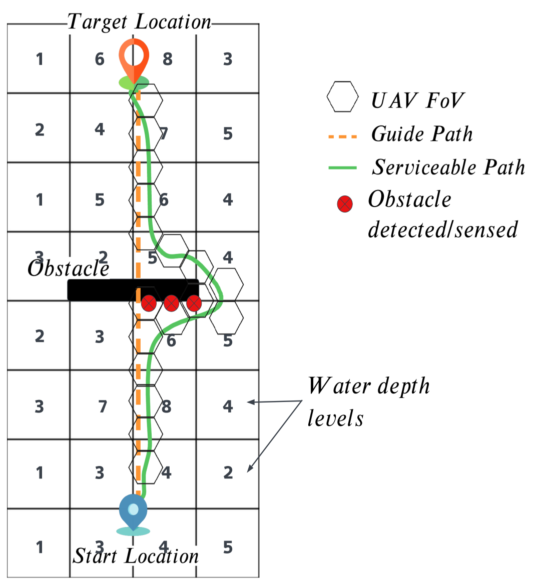


Figure 6.3: Multi-UAV coverage for serviceable path planning (depicted by green color) w.r.t. to a guided path (depicted by orange color), considering obstacles and shallow/unserviceable water regions (a cell having a water level of 3ft and less). The cell value denotes the current water level at that location in feet.

$\{s^{u_1} \times s^{u_2} \times ... \times s^{u_n}\}$. s^{u_i} : $\{u^c_{i_t}, \psi^c \rho^{u_i}_t\}$ depicts the state of the i^{th} UAV, where $u^c_{i_t}$ is the cell occupied by u_i at time t and ψ^c is the cell given by the guide path. $\psi^{u_i}_t$ is the energy level of u_i at time t. The joint action of the multi-UAV system and is given as $\{a^{u_1} \times a^{u_2} \times ... \times a^{u_n}\}$. Each UAV's action a^{u_i} lies within the feasible action set $A = \{yaw^{u_i}, pitch^{u_i}, hover^{u_i}\}$ (for more details refer to Chapter4, subsection 4.3.1). $P_{ss'}$ denotes the state transition function and s_0 describes the starting configuration of the multi-UAV system. s_{start} denotes the start location and $s_{critical}$ represents the critical location(s) to be reached. $s_{critical}$ can be a set $\{\tau_1, \tau_2, ...\tau_n\}$, if multiple locations are to be reached.

6.3.2 Multi-UAV Control Policy

To solve the MDP and achieve autonomous and continuous control of the multi-UAV system, a Deep RL method is employed, known as Multi-agent Deep Deterministic Policy Gradient (MADDPG) [60]. It works in a manner where each UAV is controlled by a separate actor-critic networks, where the actor-network is responsible for providing the feasible action based on the UAV's current state and the critic-network validates the effectiveness of the actions generated by the actor. Also, a set of target actor and critic networks is used in MADDPG to stabilize training. Critic network of a single UAV (u_i) takes the state and action of each individual UAV $(\{s^{u_1}, a^{u_1}, s^{u_2}, a^{u_2}, ...\})$ at each training time-step t into consideration to realize a state-action value function $Q^\pi(s^{u_i}_t, a^{u_i}_t)$ for UAV u_i, given as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{a_t \sim \pi} \left[\sum_{l=0}^{\Gamma} \gamma^l R_{t+l+1} | s_t, a_t \right] \tag{6.4}$$

where $0 \leq \gamma < 1$ is the discount factor (used to maintain finite sum over infinite horizon) and Γ denotes the max. time-step. The loss function used to update the critic network in reference to UAV u_i is given as ([60]):

$$L_{critic} = \frac{1}{\mathbb{B}} \sum_k \left(y_k^{u_i} - Q_k^{\mu}(s_k^{u_i}, a_k^{u_1}, a_k^{u_2}, ..., a_k^{u_n}) \right) \tag{6.5}$$

$$y_k^{u_i} = R_k + \gamma Q'(s_{k+1}^{u_i}, a_{k+1}^{u_1}, a_{k+1}^{u_2}, ..., a_{k+1}^{u_n})|_{a_{k+1}^{u_i} = \mu'(o^{u_i})} \tag{6.6}$$

where μ' denotes the target actor-network. $\mathbb{B}$ represents the size of mini-batch used for training that is sampled from the experience replay buffer and k denotes the iteration (for more details see [60]). The learning actor network then gets updated using sampled policy gradient ([60])

An estimate of the shortest path towards the critical location, provided by the guide path, prompts the UAVs to initially explore that path. The guide path is a line connecting the start and critical location corresponding to the shortest Manhattan distance between them, without any knowledge of obstacles present in the environment (the obstacles are sensed by the UAVs in real-time). The reward function $R(.,.)$ captures the objective of the system and incentivize the UAVs based on the maintained overlap and their alignment with the guided trajectory:

$$R_t(U, F) = \sum_{u_i \in U} \mathbb{T}_t(u_i, \Psi)\Lambda^c + \sum_{v_i, v_j \in F} \mathbb{O}_t(v_i, v_j) + \varphi\Lambda^c \tag{6.7}$$

$$\mathbb{T}_t(u_i, \Psi) = \begin{cases} \frac{1}{\underset{u_i \in U}{N}(u_i^c, \psi^c) + e} & if \quad \Lambda^c == 1 \\ 1 & otherwise \end{cases} \tag{6.8}$$

where $\mathbb{T}_t(.,.)$ denotes the trajectory alignment reward. ψ^c is the set denoting cells/locations covered by the guide path and u_i^c is the cell covered by the UAV u_i. $N(.,.)$ is the distance in terms of the number of cells between ψ^c and u_i^c, where the diagonally adjacent cells are at the distance of $\sqrt{2}$ and vertically/horizontally adjacent cells are at a distance of 1. e denotes a positive scalar to prevent ZeroDivisionError. Λ^c denoted whether the cell c is serviceable or not (refer Equation 6.1).

The reward received by the multi-UAV system for maintaining FoV overlap is given by $\mathbb{O}_t(v_i, v_j)$. As seen in Figure 6.2, if the euclidean distance $d(.,.)$ between the center of the FoVs of i^{th} *and* j^{th} UAV is ideal i.e. equal to D or upto allowable threshold λD, a positive reward is received by the UAVs. Whereas, a larger overlap (i.e. the distance between the center of the FoVs is less than λD) is observed or if there is a gap in FoVs' i.e. distance greater than D, a penalty is incurred by the multi-UAV system, as given in Equation 6.9.

$$\mathbb{O}_t(v_i, v_j) = \begin{cases} \beta & if\ \lambda D \leq d(v_i, v_j) \leq D \\ -\beta \times (d(v_i, v_j) - D) & if\ d(v_i, v_j) > D \\ -\beta \times (\lambda D - d(v_i, v_j)) & if\ d(v_i, v_j) < \lambda D \end{cases} \tag{6.9}$$

where β denotes a positive scalar quantity.

6.3.3 Real Time Path Planning Algorithm

A* is a heuristic algorithm [134] and one of the most widely used algorithms for path planning in a grid environment. A* determines the optimal path by sequentially expanding nodes in ascending order of their associated costs. The cost q of each sensed/reached cell r^c is generally calculated based on its distance from the start location to critical location, given as:

$$q(r^c) = v^c + g^c \tag{6.10}$$

where v^c is the heuristic distance (Manhattan distance) of the cell r^c to the critical location/cell and g^c is the distance from the start location to the reached cell (r^c) through a selected sequence of cells. The computing time required by the A* algorithm significantly relies on the number of nodes expanded. In the standard A* path planning algorithm, nodes are expanded in the order of their cost q^c (starting from minimum cost), until the critical location is reached. A* takes a large amount of space to store all possible paths and a lot of time to find them. This can be computationally very expensive and sometimes infeasible in environments will large state spaces [135]. Hence, an improved A* algorithm is proposed, known as, MEA* attaining the minimum number of expanded nodes/cells by performing a look-ahead into the search space and pruning the cells that are unserviceable.

6.3.4 Minimal Expansion A* (MEA*)

In MEA* the underlying idea is that the sensed nodes (referred to as explored cells) that are in the shallow water region are not expanded further and the nodes that are vertically/horizontally adjacent to obstacles (i.e., an obstacle is present in the region surrounding that cell) incur a cost/penalty if expanded. Also, the proposed algorithm works in real-time where the start location changes as WBV moves on the current identified path and no replanning is done. As, the environment is dynamic and the water level of the cells changes, a location deemed to be shallow earlier might become serviceable at a later stage. Such a cell could be visited again if the estimated cost of reaching the critical location from that location plus the cost of returning to that location is less than the estimated cost of going forward from the current location.
The cost function for the proposed MEA* is given as:

$$q^{MEA^*}(r^c) = v^c + g^c + o^c \tag{6.11}$$

$$o^c = \begin{cases} \alpha & if \;\; O^c == True \\ \infty & if \;\; h_c < \mathcal{W} \\ 0 & otherwise \end{cases} \tag{6.12}$$

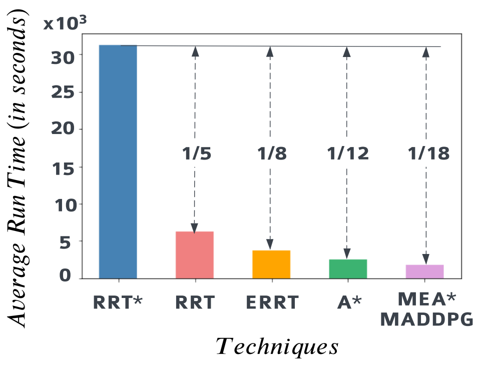
o^c is an additional cost (α) for a node/cell c that is surrounded by an obstacle. The presence of the obstacle (which is part of the guide path) and the shallow water regions/nodes (that are unserviceable) are captured by the autonomous UAVs deployed in-front of WBV for guide path updation to identify a serviceable path. An explored node is different from an expanded node, where any sensed node/cell by a UAV is said to be explored and if shallow water or obstacle is observed at that location that node is not expanded further. The expanded nodes are those that are deemed to be serviceable.

6.4 Experimentation and Results

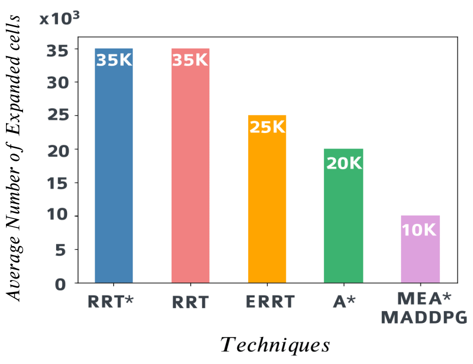
In this section, the considered experimental settings are discussed, and the performance of the proposed model MEA*MADDPG over a 2D grid is analyzed. For model implementation, Chennai city map information is encoded in a grid using the Mapbox Streets tool. Further, a 2D water layer is overlayed to simulate floods, distinguishing areas with shallow water (typically the high-elevation regions) and areas with a substantial volume of accumulated water. During experimentation the shallow water cells are processed as obstacles i.e., they cannot be a part of a serviceable path.

A team of 4 UAVs (quadrotors) is considered as part of the system during experimentation, where one of them is tasked to maintain the WBV in its FoV to provide with start location and the other three UAVs deployed in front of the WBV for autonomous sensing. The altitude of the UAVs is fixed at 20 meters above ground for all the UAVs for uniform resolution images and the UAV camera angle (half angles) is set to 45° [136] which results in an FoV area of 40 $\times$ 40 m^2. The Chennai city map having an area of 425 $\times$ 10^6 m^2 results in a grid with 515 $\times$ 515 cells where the area of each cell is equal to the FoV of a UAV. The ideal inter UAV separation distance D is set as 40 meters and the allowable overlap (λD) is 20 meters (with a value of 0.5 for ω) under which the UAVs are prone to collision. Implementation is done on Google Colab having, Intel(R) Xeon(R) CPU, 1xTesla K80 GPU, 2.30GHz CPU frequency, and 12GB RAM.

The proposed algorithm MEA*MADDPG is compared with with 4 other prevalent techniques from the literature, namely, RRT (Rapidly Exploring Random Tree) [137], RRT* [137] [134], A* [134] and real-time path planning algorithm ERRT [138]. Active Pre-training [139] is applied to MEA*MADDPG to align the UAVs with the guide path. In [137], authors provide an analytical review of path planning algorithms, namely, RRT and RRT* over a simulated environment having obstacles. In [134], authors analyzed the performance gap between A* and RRT* using a realistic simulator to handle the dynamic properties of the robot. In [138], authors perform perception-aware pathfinding for a snake robot in an unknown environment. A modified RRT algorithm known as



(a)



(b)

Figure 6.4: Performance comparison between RRT*, RRT, ERRT, A* and MEA*MADDPG over (a) average run time and (b) average number of expanded cells.

an executive rapidly-exploring random tree (ERRT) is proposed that leverages the lidar scanning-based real-time mapping for path planning.
The following performance metrics are used to evaluate the proposed model:

- Average run time
- Average number of expanded cells
- Average serviceable path length

The results are observed over 100 episodes with varying initial positions of start and critical location. The maximum run-time of a single episode is set to 50000 time-steps.

6.4.1 Performance Comparison over Average Run Time

In this experiment, the time required by each algorithm to reach the critical location is observed under equivalent environmental conditions (including start and critical locations, obstacle positioning, etc.) Figure 6.4a depicts the average recorded run time (in seconds) over 100 episodes and as seen, RRT* took approximately 5 times longer as compared to RRT and even longer as compared to other techniques. RRT performs the second worst with approximately double the run time as compared to ERRT and A*. As ERRT uses a waypoint cache and allows each new node to be selected more closely to the target waypoint (critical location), it performs better than the baseline RRT in terms of run time. However, A* and MEA*MADDPG significantly improves over the run time of RRT-based approaches. Such a difference can be intuitively analyzed as RRT and other similar approaches add new nodes randomly in the search space to find the target/critical location but have a very low probability of finding a path quickly as the process is random. Whereas, A* works on the heuristic value of the cell and expands only those nodes that have the minimum cost to reach the critical location. The proposed approach MEA*MADDPG further improves over the run time of A* i.e., approx. 1.5 times faster, as it removes the nodes that are unserviceable from further expansion by foreseeing the obstacle-bound paths and shallow water paths.

6.4.2 Performance Comparison over the Average Number of Expanded Nodes

In this experiment, the number of nodes/cells explored by each algorithm is recorded until a path to a critical location is identified. A cell that is examined by the algorithm or whose cost function is calculated is said to be explored and when that cell becomes part of a path it is said to be expanded. Figure 6.4b depicts the average number of expanded nodes/cells by each technique. As observed, RRT* and RRT have similar numbers of expanded cells. ERRT improves on the average number of expanded cells by considering path cost and selecting cells closer to the critical location/waypoint. However, the path generated by ERRT is not as optimized as A* due to its random nature. A* performs the second best in terms of expanded cells and finds the optimal path. MEA*MADDPG further improves over A* in terms of expanded cells by excluding shallow water and obstacle-surrounded cells from expansion, resulting in fewer candidate cells.

6.4.3 Performance Comparison over Average Serviceable Path Length

In this experiment, the average serviceable path length observed by each technique before reaching the critical location from the start location is observed. A serviceable path is a set of neighbouring cells connected in a sequence from the start to the critical location. Figure 6.5 depicts the performance comparison of different techniques. As observed, RRT* is able to achieve a 35% (approx.) shorter path as compared to RRT due to its rewiring operation selecting the most suitable node in terms of distance. ERRT is also able to improve over RRT as it prioritizes new nodes to be selected more closely to the critical location. However, it doesn't perform well as compared to RRT* and sees an additional 10% increase in the number of nodes in the serviceable path. A* being optimal of them all is able to discover the shortest path and on average improves upto 60% in terms of path length as compared to ERRT. MEA*MADDPG also falls short as compared to A* as it generates a longer serviceable path. Such a performance of A* is justified as it selects the least cost path after expanding all the possible paths to the critical location that could be

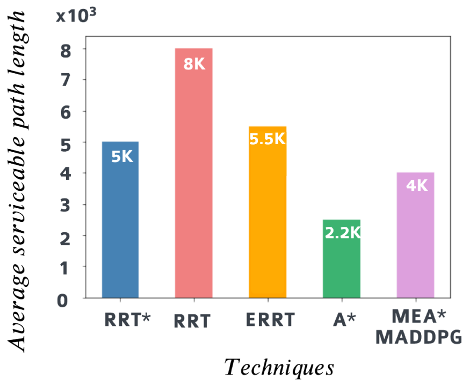


Figure 6.5: Performance comparison between RRT*, RRT, ERRT, A* and MEA*MADDPG over average serviceable path length until the critical location is reached.

better in terms of cost. But, due to this A* suffers from a worse run-time and number of expanded nodes as compared to MEA*MADDPG. Since, MEA*MADDPG doesn't do path replanning a longer serviceable path is observed by the proposed approach, but at the same time such an approach is more practical when it comes to real-world deployment.

6.4.4 Performance Comparison in Moving Obstacle Environment

In this experiment we address the problem of path planning when the environment has non-stationary obstacles. Specifically, in scenarios like flooding, abundant debris such as submerged and non-stationary objects like trees, cars, and light poles that can impede the movement of WBVs are present. To mitigate potential obstacles, the deployed UAVs conduct area coverage and sensing around the WBV. Following this aerial assessment, the path planning algorithm guides the WBV away from these detected obstacles, ensuring a safer and obstacle-free trajectory. Multiple configurations of the system are evaluated by changing the location of the WBV and the critical region to compare the performance of different algorithms. Figure 6.6 depicts a similar performance achieved by MEA*MADDPG outperforming other algorithms in case of run time and number of expanded nodes. This performance is attributed to the minimum node expansion strategy

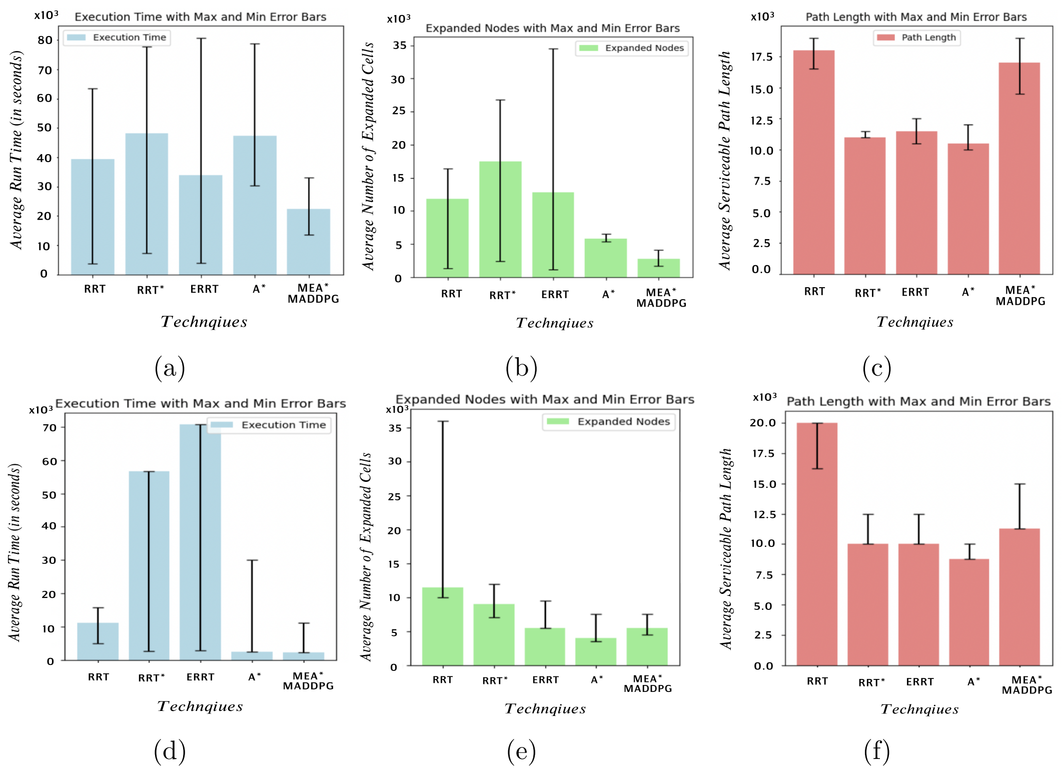


Figure 6.6: Comparative analysis in a non-stationary obstacle environment, evaluating the performance of RRT, RRT*, ERRT, A*, and MEA*MADDPG across (a) average run time, (b) average count of expanded cells, and (c) average serviceable path length until reaching the crucial location. Two different configurations are considered: (a,b,c) Start location at (5,5) and Goal location at (21,27), and (d,e,f) Start location at (19,4) and Goal location at (12,10), in a 30x30 grid.

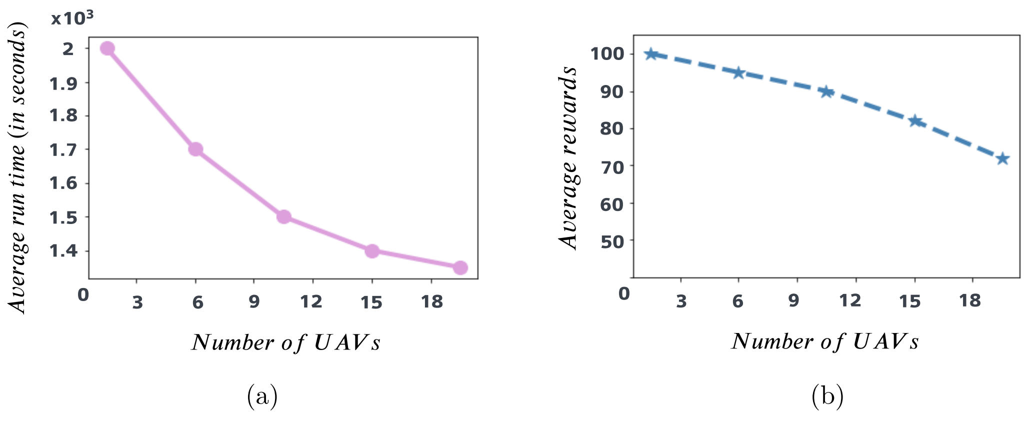


Figure 6.7: Impact of varying number of UAVs on (a) average run time of MEA*MADDPG and (b) average accumulated rewards by the multi-UAV system.

in MEA*MADDPG that prunes the unserviceable nodes to quickly identify a serviceable path to the critical location for the WBV. However, MEA*MADDPG exhibits a longer path, as also observed in case of stationary obstacles (refer to Figure 6.5). This behavior is justified as MEA*MADDPG operates in real-time, without performing replanning, which makes more sense when it comes to practical deployment of the path planning system in flood scenarios.

6.4.5 Varying Number of UAVs

In this experiment, the impact of varying number of UAVs on the run time and average rewards of MEA*MADDPG is observed. Figure 6.7a depicts the impact of scaling the number of UAVs on the model's run time. As observed, a sharp decrease in run-time is observed till the system is scaled to 10 UAVs, however, on further increase in the number of UAVs the run time seems to converge. The initial drop in run-time is easily understandable as with more UAVs the system is able to observe a larger region simultaneously which helps in realizing a serviceable path while avoiding foreseen obstacles in a brisk manner. The impact of scalability (in terms of the number of UAVs) is also analyzed w.r.t. the multi-UAV system reward in reference to Equation 6.7. Figure 6.7b depicts the change in average accumulated rewards as the number of UAVs are increased in the system in the range [1-20]. The multi-UAV model seems to suffer from penalties as it sees a continuous dip in rewards. This reflects on how the reward function is formulated. The UAV(s) are encouraged to align with the guide path but with the increase in the number of UAVs, the system suffers from overlapping incurred penalties. This suggests a maximum limit on the number of UAVs that should be included in the system to achieve optimal performance. With more UAVs, the system as a whole sees stacked incentives from optimal alignment but fails to manage the desired overlap. A reward lower than 70 is noted as poor in the current setting. As a result, a system of upto 18 UAVs is acceptable for end-to-end serviceable path planning in the considered scenario.

6.5 Chapter Summary

In this chapter, an automated path-finding method is introduced utilizing UAVs to assist WBVs in reaching critical regions during floods. The proposed algorithm MEA*MADDPG coordinates UAVs in an interleaved formation to identify serviceable regions devoid of obstacles or shallow waters, optimizing movement for WBVs. Initially, the algorithm conserves UAV energy by following a guide path and surveys adjacent regions if the current path is unserviceable. Comparative evaluations highlight the algorithm's efficacy and efficiency in swiftly generating real-time serviceable paths as compared to other prevalent techniques in the literature. Further extending the application scope, the problem of tracking a moving convoy using multiple autonomous UAVs is discussed in the next chapter.

7| Autonomous Multi-UAV Control for Moving Convoy Tracking

In this chapter, the problem of tracking a moving convoy of vehicles using autonomous UAVs is addressed. For the end-to-end coverage and tracking of the convoy, algorithms that can operate in continuous state and action spaces are required. Actor-critic methods [19] in reinforcement learning are well-suited to handle such continuous domain problems. Hence, for this tracking application, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [60] algorithm is employed to enable the UAVs to maintain the moving convoy in their joint Field-of-View (FoV). However, the networks in MADDPG are randomly initialized which leads to delayed convergence and require a substantial number of samples [40] to train. The poorly defined target functions also introduce overestimation bias in the policy, causing the policy to diverge [34]. To tackle this effectively, a novel target value function estimator is proposed leveraging the observed information, i.e., the convoy's current trajectory or traveled path. This observed data is used to compute the covariance matrix of a Gaussian process regression (GPR) model [140]. This GPR model is then used as the target critic in the MADDPG model. Further, the complexity of the convoy trajectories is encoded in the kernel function of the GPR to keep the value estimates within bounds. In addition to this, the reward function is formulated based on the tracking and overlapping incentives derived from the positions of the UAVs and the convoy. This helps the UAVs to keep the vehicles of the convoy near the center of their FoVs resulting in higher coverage (assuming each UAV is tasked to track a single vehicle of the convoy). The proposed approach is evaluated over different road trajectories with varying degrees of complexity. Further, the model is also tested using Gazebo [55] simulator that has a real-world physics engine.
The rest of the chapter is organized as follows. section 7.1 presents the overall system description along with the system's objective. In section 7.2, the proposed GPR-MADDPG algorithm is discussed, highlighting overestimation bias problem, GPR modelling and kernel function. section 7.3 discusses the experiments and results. Finally, in section 7.4, a concise summary of the chapter is provided.

7.1 System Model

In the system model, a group of n UAVs $U = \{u_i | i \in \{1, 2, ..., n\}\}$ is considered tasked to track a convoy of vehicles moving on the ground (for simplicity an equal number of UAVs and vehicles are considered, although the number of vehicles can be larger). Each UAV is tasked to track a single vehicle of the convoy and try to keep it at the center of its FoV. The joint FoV of all the UAVs is given by the set $F = \{v_i | i \in \{1, 2, ..., n\}\}$, where, $v_i = (x_{u_i}, y_{u_i})$ denoting the center of the FoV. The overall objective of the system

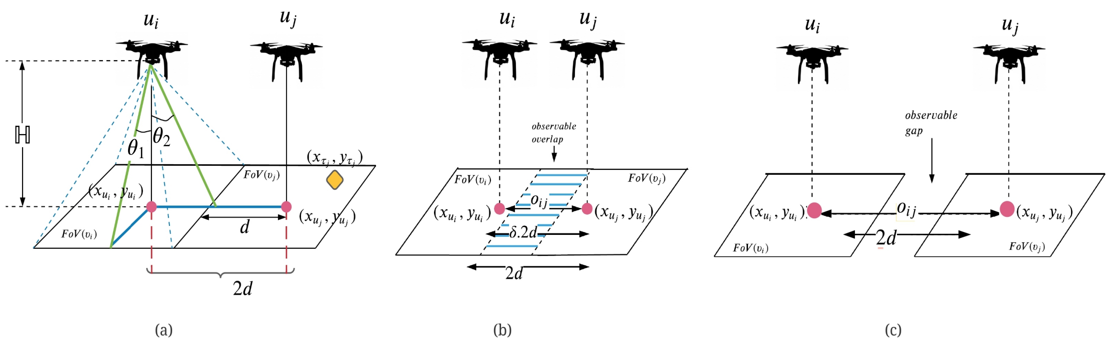


Figure 7.1: Overlapping of FoVs for a pair of UAVs (u_i, u_j) (a) Optimal case of Overlap (b) Overlap beyond given threshold (c) Disjoint FoVs with no overlap.

is to keep track of the convoy with each individual UAV tracking a particular vehicle τ_i $(T = \{\tau_i | i \in \{1, 2, ..., n\}\})$ within its FoV, ideally at its center. The overall objective of the system is given as:

$$\max \sum_{t=0}^{\infty} \left(\sum_{\tau_i \in T} \mathbb{T}_t(\tau_i, F) + \sum_{\upsilon_i, \upsilon_j \in F, i \neq j} \mathbb{O}_t(\upsilon_i, \upsilon_j) \right) \tag{7.1}$$

where t is the time. $\mathbb{T}_t(.,.)$ represents the incentive associated with tracking error based on the UAV's position and the relative position of the vehicle it is tracking and $\mathbb{O}_t(.,.)$ represents the overlapping incentive conditioned on the maximum allowable overlap.

$$\mathbb{T}_t(\tau_i, F) = \begin{cases} \frac{1}{\min\limits_{\upsilon_j \in F} d(\tau_i, \upsilon_j) + d'} & if\ \mathbb{1}_t(\tau_i) = 1 \\ -e & otherwise \end{cases} \tag{7.2}$$

where e, d' are positive scalar quantities. min $d(\tau_i, \upsilon_j)$ finds the distance from the 2D projected position of the UAV to the vehicle being tracked (d' is a small positive scalar to avoid ZeroDivisionError). $\mathbb{1}_t(\tau_i)$ is an indicator function which denotes whether the target vehicle τ_i is present in the joint FoV or not. The multi-UAV system obtains incentives corresponding to the degree to which the UAV is perpendicularly aligned with the target along with end-to-end coverage of the whole convoy. If the altitude of the UAV is $\mathbb{H}$ meters, the FoV of the UAV is $(2h \times tan(\theta_1))(2h \times tan(\theta_2))$ m^2 w.r.t UAV angles (as seen in Figure 7.1(a)). A fixed altitude for all the UAVs is considered for equal resolution of the joint FoV.

Let the center of FoV for u_i be (x_{u_i}, y_{u_i}) and for u_j as (x_{u_j}, y_{u_j}) where, (x, y) denotes the coordinates of the 2D projected position of the UAV. As the UAVs follow an ordered formation, the i^{th} UAV has an overlap of FoVs with the $(i+1)^{th}$ UAV. In the ideal case with edge-to-edge FoV coverage without any gaps the overlapping distance between u_i and u_j (where $j = i + 1$) is $o_{i,j} = 2d$ (see Figure 7.1(a)) where, $o_{i,j}$ is the euclidean distance between the center of the FoV's (or the 2D projected positions) of u_i and u_j. However, as the environment is noisy, some threshold overlap $(\delta.2d)$ is allowed (refer Equation 7.3).

Beyond the threshold, as shown in Figure 7.1(b), the overlapping incentive is reduced according to Equation 7.3 second condition. Figure 7.1(c) shows the case of disjoint FoVs. In this case, the non-overlapping distance between the FoVs (υ_i, υ_j) is calculated and the overlapping incentive is reduced according to Equation 7.3 third condition.

$$\mathbb{O}_t(\upsilon_i, \upsilon_j) = \begin{cases} c & if\ \delta.2d \leq o^t_{i,i+1} \leq 2d \\ -c \times g(\delta.2d - o^t_{i,i+1}) & if\ \ o^t_{i,i+1} < \delta.2d \\ -c \times g(o^t_{i,i+1} - 2d) & if\ \ o^t_{i,i+1} > 2d \end{cases} \tag{7.3}$$

In the above equation, c and δ are positive scalars. The value of δ lies between (0,1). $2d$ is the ideal distance between two FoVs with no gaps. Function $g(.)$ is used to scale down the value between [0,1] for mathematical simplification. A target (vehicle) τ_j is said to be tracked by the UAV u_j (as seen in Figure 7.1(a)), if the following conditions are satisfied:

$$\begin{aligned} x_{u_j} - h \times tan(\theta_1) \leq x_{\tau_j} \leq x_{u_j} + h \times tan(\theta_1) \\ y_{u_j} - h \times tan(\theta_2) \leq y_{\tau_j} \leq y_{u_j} + h \times tan(\theta_2) \end{aligned} \tag{7.4}$$

where, (x_{τ_j}, y_{τ_j}) are the coordinates of the vehicle τ_j being tracked in u_j's FoV. In the event of a collision between the UAVs at any point in time, the episode is terminated, and a new one begins.

7.2 Proposed Methodology

In this section, the discussion centers on the proposed solution for multi-UAV based convoy tracking, emphasizing the target value function approximation achieved using GPR-based target critic.

7.2.1 MDP Formulation

A Multi-agent Reinforcement Learning (MARL) model is described by a Markov Decision Process (MDP) quintuple $< S, A, P_{ss'}, R, s_0 >$, where S is the joint state space of the n UAVs: $\{s_1 \times s_2, ... \times s_n\}$, where s_i denotes the location of UAV u_i and the relative location of the vehicle it is tracking $[x_{u_i}, y_{u_i}, x_{\tau_i}, y_{\tau_i}]$ (value of (x_{τ_i}, y_{τ_i}) may be unavailable based on whether the vehicle is present in the FoV of u_i or not). a_t denotes the joint action of the UAVs: $\{a_1 \times a_2, ... \times a_n\}$ at time t, where a_i belongs to a feasible action set $A : [yaw_i, pitch_i]$. $P_{ss'}$ denotes the model of the environment. R is the reward function and s_0 denotes the starting configuration of the UAVs.

In this work, the objective is to learn a joint policy in order to maximize the cumulative rewards received by the multi-UAV system in the long run. Policy $\pi(s_t)$ provides the actions for all the UAVs present in the common state-space at any given time t. Further, for a given policy (π) a state-action value function $Q^\pi(s_t, a_t)$ defines the long-term desirability of a state-action pair given as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{a_t \sim \pi}\left[\sum_{l=0}^{\infty} \gamma^l R_{t+l+1} | s_t, a_t\right] \tag{7.5}$$

where $0 \leq \gamma < 1$ is the discount factor.

The reward function $R(.,.)$ is formulated based on the system's objective (Equation 7.1), given as:

$$R_t(T, F) = \sum_{\tau_i \in T} \mathbb{I}_t(\tau_i, F) \quad + \sum_{\upsilon_i, \upsilon_j \in F, i \neq j} \mathbb{O}_t(\upsilon_i, \upsilon_j)) \tag{7.6}$$

The value function $Q(s, a)$ represents the expected return of all the future rewards from a given state-action configuration. Function approximation methods in Deep RL work by updating this value function with an exploration-exploitation trade-off [141] to learn an optimal policy. The motivation behind proposing GPR based target for value function approximation is to generalize the value function over the state-action space based on the observed data and maintain the estimates within bounds. The GPR approximation is implemented individually for each of the UAVs within the system.

7.2.2 GPR modelling

GPR works as a function approximation technique that provides us with a continuous estimate of the value function along with the variance over these estimates. GPR technique has earlier been adopted to learn continuous state-action value function in SARSA(λ) (see Engel et al. [142]). GPR has also been applied previously along with dynamic programming to provide value function estimates over unobserved locations (see Deisenroth et al. [143]). In the GPR model, a prior is placed directly over the value function and conditioned on observations (state-action pairs) to provide Q-value estimates.

In the considered scenario, the observed state-action pairs are denoted as $M_t = [m_i]_{i=\beta}^{t}$, where, $m_t = [s_t, a_t]$ and t is the time-step upto which observations have been made. β limits the number of previous time-steps considered for formulating the covariance matrix. Hence, for a new input $m_{t+1} = [s_{t+1}, a_{t+1}]$, the target Q-value $(Q'_{GPR}(t+1))$ is the estimated GPR mean calculated using kernel function (k), covariance matrix $C = C(M_t, M_t)$ and additive white noise $\mathcal{N}(0, \sigma_n^2)$.

$$\begin{gathered} Q'_{GPR}(t+1) = C(m_{t+1}, M_t)[C(M_t, M_t) + \sigma_n^2 I]^{-1} Z \\ Z = \left[Q'_{GPR}(i)\right]_{i=\beta}^{t} \end{gathered} \tag{7.7}$$

The kernel function using an alignment parameter (α_p) is given as:"

$$
\begin{aligned}
k(m_t, m_{t+1}) &= exp\left(-||m_t - m_{t+1}||^2\right) \times \alpha_p \\
\alpha_p &= \psi \times \left(\frac{m_t \times m_{t+1}}{||m_t|| \times ||m_{t+1}||}\right)
\end{aligned} \tag{7.8}
$$

where ψ is a positive scalar in range (0,1]. α_p, is the alignment incentive based on the angle between the observed point at time t and the current input at time $t+1$. The underlying idea behind the proposed GPR is that the state-action configurations that are closer to each other with less angular deviation have similar Q-values. This intuition is derived based on the underlying application of tracking the moving convoy. The mean Q-value from the GPR model is then used as the target to calculate the critic's loss. This mean Q-value is scaled using the alignment parameter α_p based on the alignment of the new inputs with respect to the previously observed points.

Overestimation Bias

The existence of overestimation bias is due to function approximation errors in actor-critic methods as the policy is updated w.r.t. an approximate critic [34]. The actor-network is updated based on the state-action value estimated by this critic network resulting in sub-optimal policies. The critic network parameters are updated using the gradients from a biased target that can be noisy. Sometimes these noisy and biased estimates overshoot the true target value ($\mathbb{E}\left[Q_\theta(s,a)\right] \geq \mathbb{E}\left[Q^\pi(s,a)\right]$). Hence, the model keeps on trying to maximize over a false estimate. This overestimation develops into a more prominent bias over many iterations if left unnoticed. The alignment parameter (α_p) regulates the Q-value estimates within bounds when high uncertainty in the convoy trajectory is observed. As the MADDPG model suffers from overestimation bias in the Q-value estimates [34], the scaling of the Q-value estimates using the alignment parameter helps to improve the model by limiting the extent of overestimation bias.

7.2.3 The Proposed GPR-MADDPG Model

The GPR model is suited for the tracking application as the kernel function of the GPR is adapted using the tacking incentive based on the convoy movement. This tracking incentive can be intuitively seen as a function that is normally distributed along the state-action space. Thus, we can find the continuous estimate of target Q-values in the state-action space by using this incentive as a measure in the kernel function of GPR. Thus, in the proposed algorithm GPR-MADDPG, the GPR is used as a target value function approximator for the MADDPG model to learn the UAV policy $\pi(s_t)$. Figure 7.2 shows the overall architecture of the GPR-MADDPG model, and as illustrated, each UAV has an actor and critic network along with a GPR target value estimator. The output of GPR is used to calculate the critic's loss for training the critic network.

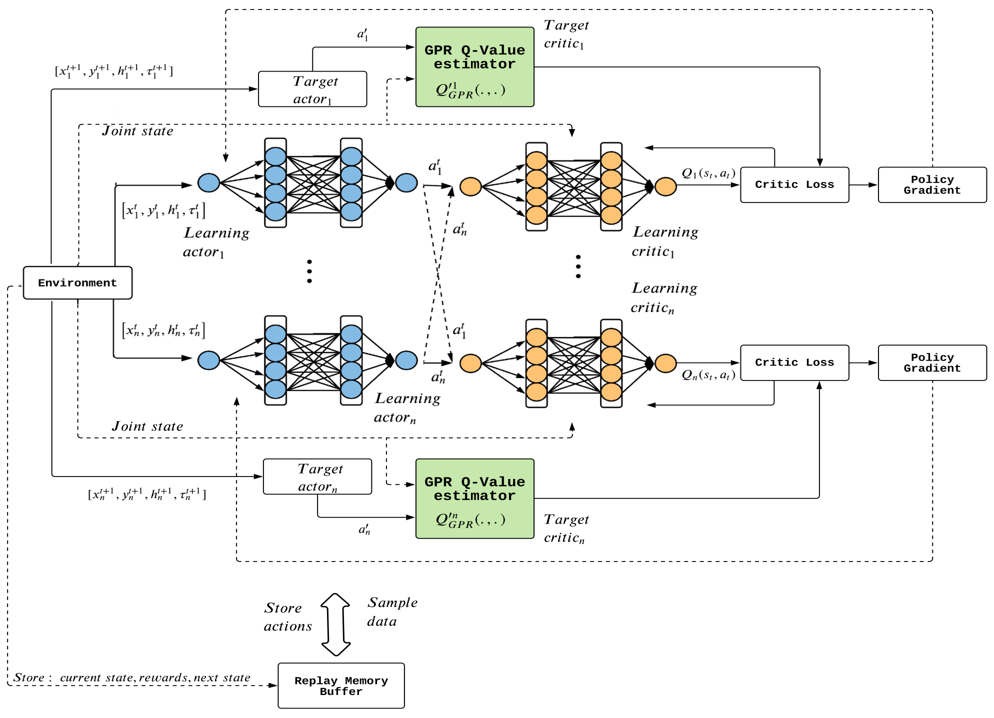


Figure 7.2: Illustration of GPR-MADDPG architecture.

The loss function used to update the critic network is given as:

$$L_{critic} = \frac{1}{\mathcal{B}} \sum_t (y_t - Q(s_t, a_t|\theta^Q))^2 \tag{7.9}$$

where θ^Q are the parameters of the learning critic network and $\mathcal{B}$ is mini-batch size.

$$y_t = R_t + \gamma Q'_{GPR}(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})) \tag{7.10}$$

where μ' represents the target actor network. Unlike the MADDPG algorithm, the proposed model updates the critic network parameters based on GPR estimated target Q-values(Q'_{GPR}).

The learning actor network then gets updated using sampled policy gradient:

$$\nabla_{\theta^\pi} J \approx \frac{1}{\mathcal{B}} \sum_t \nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\pi(s_t)} \nabla_{\theta^\pi} \mu(s|\theta^\pi)|_{s_t} \tag{7.11}$$

The proposed GPR-MADDPG algorithm is given in algorithm 4. As depicted, the system-related hyperparameters such as the number of target vehicles (n), UAV altitude ($\mathbb{H}$), value of γ, number of time-steps Γ, etc., are initialized. Further, at each time-step t within an episode the joint action A_t is executed by the UAVs based on their joint state S_t. Next, the reward received by the UAVs and their next state as perceived from the environment is observed. Tuple (S_t, A_t, R_t, S_{t+1}) is stored in the replay buffer $\mathcal{Z}$. After initial exploration and experience gathering, the multi-UAV model is now trained

using the proposed GPR-MADDPG algorithm. Random mini-batches of experiences is extracted from the buffer and used to realize the target Q-value $Q'_{GPR}(.,.)$ based on state s_{t+1} and the action received from target actor $\mu'(s_{t+1}|\theta^{\mu'})$. Next, critic loss is calculated using the learning critic Q-value $Q(s_t, a_t|\theta^Q)$ and target value y_t, where y_t calculated using current reward R_t and target critic Q-value as given by Equation 7.9 and Equation 7.10. To learn the optimal policy the learning actor weights are updated in the direction of the gradient, as given in Equation 7.11.
The computation cost of the GPR-MADDPG algorithm as seen from the algorithm is $O(E * \Gamma * S)$, where E is the number of episodes, Γ is the max_time_step of each episode and S denotes the number of states at any given time-step.

7.2.4 GPR-MADDPG Convergence

As the actor-critic model [144] is gradient-based, in practicality, it's difficult for such models (i.e., GPR-MADDPG in this case) to attain a global optimal policy, but for GPR-MADDPG model to converge, the L_{critic} should be under the desired threshold and should not increase as the number of episodes tends to infinity.

1. At any given time(t), the Q-value given by current policy (Q^π) is some approximation of optimal Q-value (Q^*).

2. The current approximation us updated in a greedy manner to reduce $d(\tau_i, v_j)\ \forall i \in T\ \forall j \in F$.

$$\begin{split}|Q^*(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) - Q^{\pi'}(s_t, \mu'(s))| \leq |Q^*(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) - \\ Q^{\pi|a}(s_t, \mu(s))|\end{split} \tag{7.12}$$

where, π' is the improved approximation of previous policy π for a given state s [19].

But, the knowledge of Q^* is not available, so the model uses the approximated Q value estimated from the GPR target. In reference to the Equation 7.8:

$$k(m_t, m_{t+1}) = exp\left(-||m_t - m_{t+1}||^2\right) \times \alpha_p$$
$$\alpha_p = \psi \times \left(\frac{m_t \times m_{t+1}}{||m_t|| \times ||m_{t+1}||}\right)$$

The covariance matrix (C) produced by applying the kernel function on the observed pairs will specify the statistical relationship between them. The optimal Q value (Q^*) will be given by the policy that achieves the highest global reward (R) possible and in ideal terms would be achieved at the position, where:

$$d(\tau_i, v_j) = 0 \tag{7.13}$$

In reference to Equation 7.6:

$$R_t = \sum_{\tau_i \in T} I_t(\tau_i, F) + \sum_{\upsilon_i, \upsilon_j \in F, i \neq j} O_t(\upsilon_i, \upsilon_j)$$

In reference to Equation 7.2:

$$I_t(\tau_i, F) = \begin{cases} \frac{1}{\min\limits_{\upsilon_j \in F} d(\tau_i, \upsilon_j) + d'} & if\ 1_t(\tau_i) = 1 \\ -e & otherwise \end{cases}$$

At, $d(\tau_i, \upsilon_j) = 0$, we have $m_t = m_{t+1}$,
At, $m_t = m_{t+1}$, the kernel function gives output as 1 (the maximum correlation that could be achieved).

So at optimality,

$$Q^*_{GPR}(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) \equiv Q^*(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) \tag{7.14}$$

Having the updated policy (π'), such that:

$$\begin{aligned} & d(t_i, v_j)^{\pi'} < d(t_i, v_j)^{\pi} \\ & \implies Q^{\pi'}_{GPR}(s_{t+1}, \mu(s_{t+1}|\theta^{\mu'})) < Q^{\pi}_{GPR}(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})) \end{aligned} \tag{7.15}$$

That implies,

$$\begin{aligned} |Q^*_{GPR}(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) - Q^{\pi'}(s_t, \mu'(s))| \leq |Q^*_{GPR}(s_{t+1}, \mu^*(s_{t+1}|\theta^{\mu^*})) - \\ Q^{\pi}(s_t, \mu(s))| \end{aligned} \tag{7.16}$$

And,

$$\begin{aligned} & as\ \ t \to \infty \\ & Q^{\pi^t}_{GPR}(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})) \to Q^*_{GPR}(s_{t+1}, \mu'(s_{t+1}|\theta^{u^*})) \end{aligned} \tag{7.17}$$

7.2.5 Assumptions and Limitations

In the proposed approach the focus on maintaining a single vehicle at the center of the FoV of each UAV, however, this can be easily extended to scenarios involving multiple vehicles within each FoV by adapting the reward function to multiple vehicles. Is it assumed that the environment is markovian that follows a stationary distribution. This assumption is a necessary condition for the convergence of the UAVs' policies. The critic networks have access to the joint observations comprising the states and actions of every UAV during training (as in the case of standard MADDPG algorithm [60]). This helps in addressing the

Algorithm 4: GPR-MADDPG algorithm

```
1  Input: Number of moving vehicles ($n$), UAV altitude ($\mathbb{H}$), $\gamma$, $\Gamma$
2  for episode=1,2,... do
3      Initialise the state of the target vehicles in the center of the FoV of the
        assigned UAVs
4      Set time($t$)=0
5      while $t \leq \Gamma$ do
6          Observe joint state $S_t$ (refer subsection 7.2.1)
7          For each agent $i$, select an action $a_i$ w.r.t. current policy $\pi$
8          Execute joint action $A_t : \{a_1 \times a_2, ..... \times a_n\}$
9          Observe joint Reward $R$ and next state of each agent $S_{t+1}$ (refer
            subsection 7.2.1)
10         Append replay buffer $\mathcal{Z} \leftarrow (S_t, a_t, R_t, S_{t+1})$
11         $S_t \leftarrow S_{t+1}$
12         for each agent do
13             Sample a random mini-batch of $\mathcal{B}$ samples $(S_k, a_k, R_k, S_{k+1})$ from $\mathcal{Z}$
14             Set $y_k = R_k + \gamma Q'_{GPR}(S_{k+1}, \mu'(S_{k+1}|\theta^{\mu'}))$ (given by Equation 7.10)
15             Update critic by minimizing the loss (given by Equation 7.9):
16                 $$L_{critic} = \frac{1}{\mathcal{B}} \sum_k (y_k - Q(s_k, a_k|\theta^Q))^2$$
17             Update actor network using the sampled policy gradient (given by
                Equation 7.11):
18                 $$\nabla_{\theta^\pi} J \approx \frac{1}{\mathcal{B}} \sum_k \nabla_a Q(s, a|\theta^Q)|_{s=s_k, a=\pi(s_k)} \nabla_{\theta^\pi} \mu(s|\theta^\pi)|_{s_k}$$
19         end
20         Update joint policy $\pi$
21     end
22 end
23 return Updated joint policy $\pi$
```

non-stationarity w.r.t. to multiple UAVs. The altitude of the UAVs is fixed at $\mathbb{H}$ so that the FoV images can be easily stitched together with same ground resolution. The system model does not account for the energy levels of the UAVs. As a result, the assumption is made that the UAVs' batteries are replaced and managed at the ground control station.

7.3 Experimentation and Results

In this section, the performance of the proposed GPR-MADDPG algorithm is compared with various baseline approaches,namely MADDPG [60], MA-A2C [145] and MA-A3C [146]. The adaption of MA-A2C and MA-A3C algorithms is in reference to the cited literature, where a centralised critic is applied conditioned on the state (i.e., shared by all the UAVs) of the environment rather than the individual history of observations. Additionally, the scalability of GPR-MADDPG concerning the number of UAVs in the

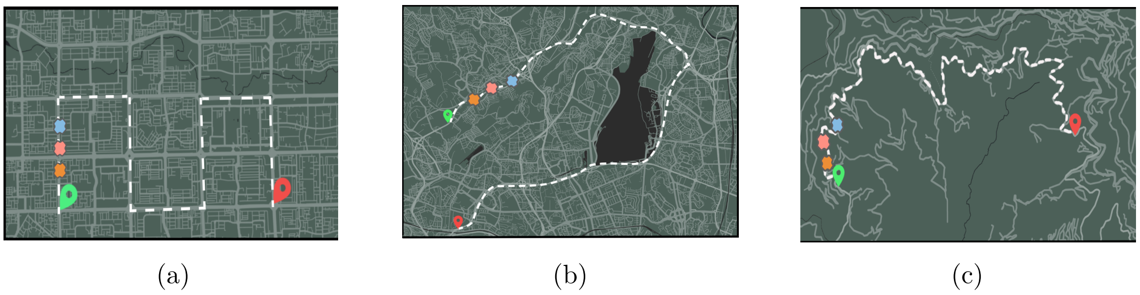


Figure 7.3: Real-world road networks, where, the green marker represents the start of the convoy course and the red marker represents the end. The white dotted line is denoting the path taken by the convoy (nodes in orange, pink and blue represent the 3 vehicles of the moving convoy). (a) Straight track with 90° turns (b) Curved track (c) Switchback track.

system is also addressed. GPR-MADDPG is also compared with three recent techniques from literature viz. Meta-TD3 [79], TF-DQN [80] and A* [147] that are proposed for UAV based object tracking applications. Three distinct road networks are selected, namely, Straight track with right angle turns (Figure 7.3(a)), Curved track (Figure 7.3(b)) and Switchback track (Figure 7.3(c)). There are two variants of convoy movement opted, viz. Constant velocity convoy and Varying velocity convoy. Three vehicles in the convoy are considered while performing the experiments with the scalability experiment performed separately.

In all the experiments, the actor network learning rate is set to 1e^{-4} and the critic network learning rate to 1e^{-3} [32]. Each episode runs for 1000 steps. FoV overlap parameter (δ) is set to 0.75. The maximum velocity for UAV is set as 40 km/hr [148] and the rate of change of velocity is set to $2m/s^2$. For simulating the environment, OSRM tool [54] is used to collect latitude-longitude coordinates for varied road tracks over which the convoy movement is tracked. Dataset used can be found in the given GitHub repository [1]. Results are observed over 5 different random seeds and their average is highlighted in the plots to analyze the statistical significance of the proposed model and its generalization ability.

7.3.1 Convoy Travelling at Constant Velocity

In this experiment, the speed of the convoy is set to 35 km/hr. Figure 7.4(a), Figure 7.4(b) and Figure 7.4(c) plot the average rewards per episode for the considered tracks. These plots compare the performance of GPR-MADDPG against MA-A2C, MA-A3C and MADDPG. As can be observed in these figures, the proposed GPR-MADDPG model outperforms all other baseline approaches on all three different tracks used in the experimentation. It can be noted from Figure 7.4, MA-A3C and MA-A2C have almost similar performance with not much improvement in rewards over the period of 10000 episodes. Both these algorithms are very sensitive to small changes in hyperparameters and difficult to train in highly dynamic and continuous environments. Looking at MADDPG and GPR-MADDPG, both are able to learn and improve as episodes go by but higher

[1]https://github.com/Armaan-Garg/Dataset-GPR-MADDPG

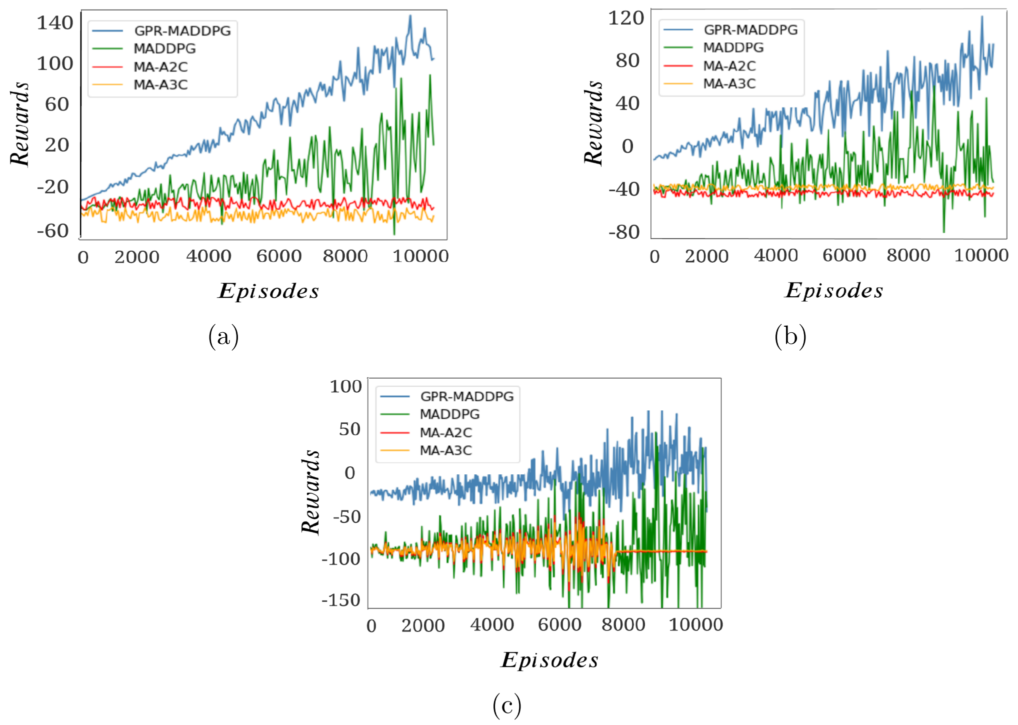


Figure 7.4: Performance comparison of GPR-MADDPG, MADDPG, MA-A2C and MA-A3C methods corresponding to (a) Straight track (b) Curved track (c) Switchback track, where the convoy is travelling at constant velocity.

fluctuation in rewards is observed in the case of MADDPG relative to GPR-MADDPG. This corresponds to the fact that utilizing the proposed kernel function in GPR-MADDPG limits fluctuations, by avoiding the overestimation of Q-values and facilitating the learning of a more stable policy.

The complexity of the tracks can be deduced from Figure 7.4, as in the case of the curved track there is almost 15 percent drop in observed rewards as compared to the straight line track. These rewards further diminish by approximately 15 percent in the case of switchback track. Higher fluctuation in the case of curved and switchback tracks indicates a relatively unstable policy as to the one learnt in the case of straight-line track. One of the key differences between the results for curved and switchback tracks is that in the case of switchback there is no significant improvement in results almost halfway through the total episodes. A rise in rewards is observed only after 6000 episodes. Also, a very low rise in rewards is observed contrary to the linear rise in the case of curved track.

7.3.2 Convoy Travelling at Varying Velocity

This section presents the results gathered from the experiment where the speed of the vehicles in the convoy is varied in the range of 5 km/hr to 35 km/hr. In addition, pre-assigned vehicle stops (randomly generated) were introduced along the path to further test the robustness of the proposed model. The plots in Figure 7.5 depict the results corresponding to the three tracks explained earlier. A significant drop in the performance of all the algorithms can be observed in the results due to the additional complexity

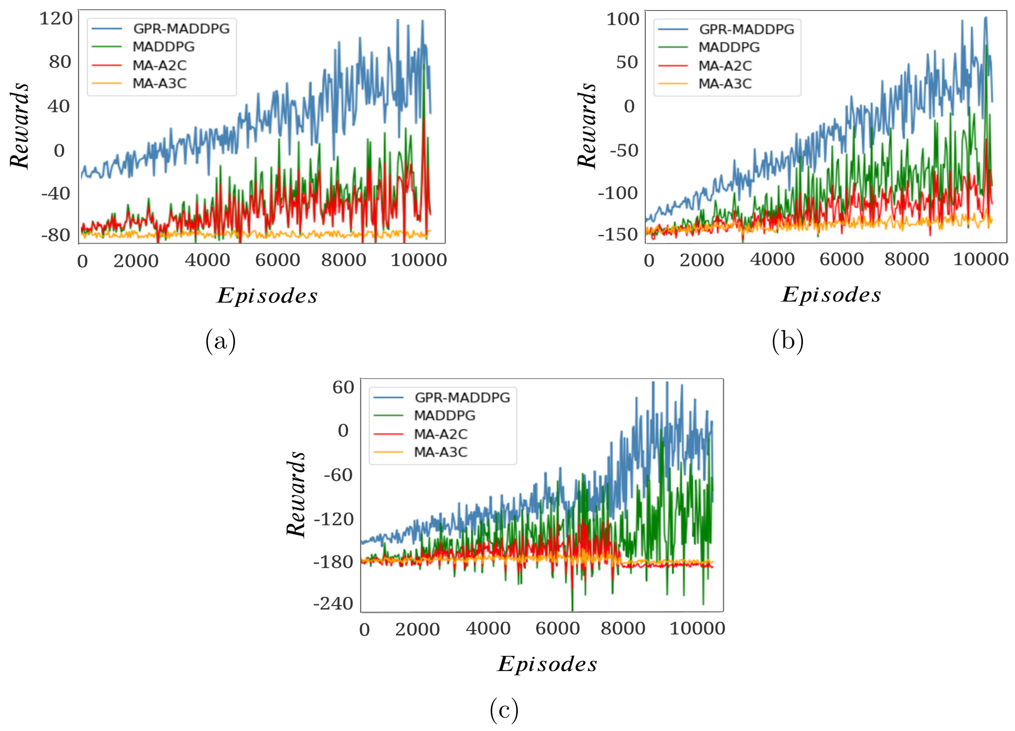
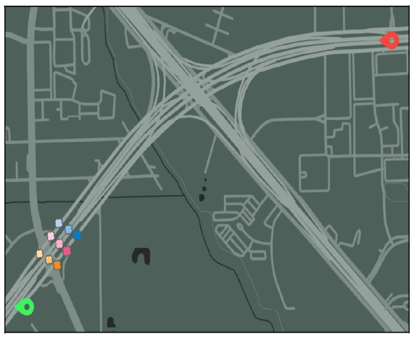


Figure 7.5: Performance comparison of GPR-MADDPG, MADDPG, MA-A2C and MA-A3C methods corresponding to (a) Straight track (b) Curved track (c) Switchback track, where the convoy is travelling at varying velocity.

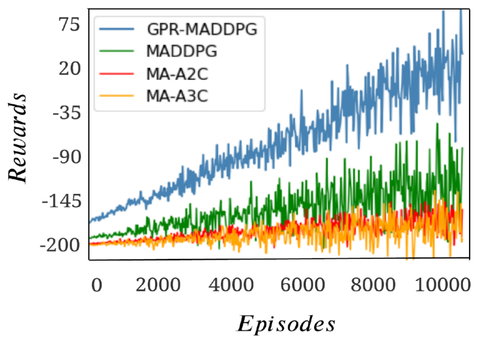
of varying velocity of vehicles. MA-A2C seems to perform better than MA-A3C and is equivalent to MADDPG to some degree. MA-A3C consists of UAVs that interact with a different copy of the environment in parallel for better exploration, but in this scenario, it just hinders with the overall performance of the multi-UAV system as it performs the worst. All the algorithms show higher fluctuation in overall rewards depicting the complexity of tracking in varying velocity environments. However, as can be noted, the proposed GPR-MADDPG still outperforms the baseline approaches, demonstrating a more effectively trained policy for the UAVs. This can be attributed to the fact that the GPR based model is able to improve the actor-network by using appropriate target values and limits the negative rewards in the long run. GPR-MADDPG shows fluctuations relative to MADDPG (especially in the case of switchback track) highlighting the extreme difficulty in tracking the convoy for the multi-UAV system in such conditions.

7.3.3 Convoy Travelling on Multi-lane Track at Varying Velocity

In this experiment, the multi-UAV based tracking is subjected to a different formation of convoy where vehicles are travelling in multiple lanes in 3x3 formation (Figure 7.6(a)) at a varying velocity between 5 - 35 km/hr. The management of FoV overlaps is more crucial in this particular setting and could incur higher negative rewards. The plot in Figure 7.6(b)) depicts the cumulative rewards accumulated by various models. As can be seen, the proposed model performs significantly well as compared to other techniques in this multi-lane setting. The better performance of the proposed GPR-MADDPG model



(a)



(b)

Figure 7.6: (a) Real-world road network of multi-lane track (b) Performance comparison of GPR-MADDPG, MADDPG, MA-A3C and MA-A2C methods over a multi-lane track.

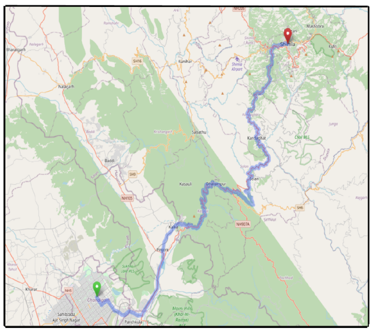
can be attributed to the modelling of tracking and overlap objectives using the kernel function and reward function. This encoding of the specific attributes of the tracking application into the Q-value estimates results in its good performance. A similar trend in performance is observed with GPR-MADDPG performing the best followed by MADDPG. MA-A2C and MA-A3C perform almost equivalently with MA-A2C outperforming the latter in a few episodes. The accumulated rewards are almost similar to the one where targets are travelling on a curved track with varying velocity, but in this scenario, there is a lower overall reward due to a larger number of overlaps between UAVs.

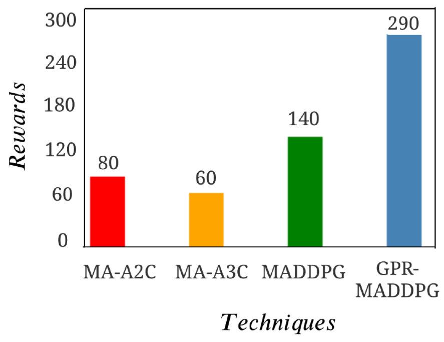
7.3.4 Unknown Test Trajectory

To test the policy generated by the GPR-MADDPG over unseen trajectories, a test trajectory is selected with a route length of 100 km (Figure 7.7(a)). The convoy is travelling at a varying velocity between 5 - 35 km/hr. The models in this experiment do not learn or update their policy, instead, the policies learnt in the case of switchback track with varying velocity is used to track the convoy over this entire stretch. The observed rewards over a single episode is reported. This experiment evaluates the robustness of the learnt policies over a longer duration while covering any form of track that the convoy might observe in the real-world. As can be observed from the plot in Figure 7.7(b), the policy learnt using GPR-MADDPG produces better rewards over the long route followed by the performance of the MADDPG model. This experiment shows that the proposed model is able to tackle any form of track with relative ease as compared to other baseline approaches. This type of track poses different challenges corresponding to the road structure over which the target vehicles are travelling. An accumulation of positive rewards in case of all the different models highlights the effectiveness of Deep RL based techniques w.r.t. autonomous robots in unknown environments.

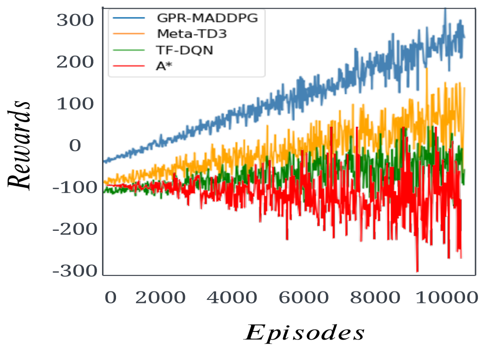
7.3.5 Comparison with State-of-the-art Models

In this section, the proposed GPR-MADDPG algorithm is compared with three recent algorithms from the literature namely Meta-TD3 [79], TF-DQN [80] and A* [147]. In


(a)



(b)



(c)

Figure 7.7: (a) Real-world road network of 100 km stretch (Test trajectory) (b) Performance comparison of GPR-MADDPG, MADDPG, MA-A3C and MA-A2C methods over the test trajectory. (c) Performance comparison between GPR-MADDPG, Meta-TD3, TF-DQN and A* over a 100 km stretch.

[79], the authors have proposed an algorithm that enables the optimization of the initial parameters of a MARL model to improve generalization ability in uncertain environments. To adapt this method to the current setting the learnt Meta-TD3 policy is trained on each track, straight track, curved track and switchback track (in this particular order) and the experience gained while training over the straight track is carried over as part of the replay buffer when training over the curved track and similarly when training next on the switchback track. Another algorithm viz. TF-DQN [80] is a Deep Q-Network based approach to persistently track a dynamic target using UAV along with obstacle avoidance. This technique is adopted as such with each UAV using individual TF-DQN algorithm running at their end, but the action space is discretized by considering the *yaw* and *pitch* action values in the range [-1,1,0.1], where 0.1 step denotes 18-degree (approx.) angle shift. GPR-MADDPG and TF-DQN utilize their policies learnt over the switchback track. Additionally, selecting a heuristic-based approach from the literature [147] acts as a baseline for other Deep RL based algorithms. The A* search algorithm calculates the cost of all the next positions that may be reached from the current position using the cost function (this function is calculated based on the objective function given in this chapter) and adds the least cost position to the trajectory. Based on the current state and speed of the UAV and using discretized action space as in TF-DQN, the possible next positions are identified. In such a manner the A* is adopted and implemented for comparison. Also,

each UAV runs a separate A* algorithm at their end.

All three algorithms are compared under identical environmental settings. In this experiment, the test trajectory is used to evaluate the algorithms. Results in Figure 7.7(c) depict the average rewards over 10000 episodes attained by each model. A similar level of performance is observed in the case of TF-DQN and Meta-TD3 in terms of rewards with Meta-TD3 slightly outperforming the TF-DQN based model. GPR-MADDPG performs the best, outperforming both Meta-TD3 and TF-DQN throughout the episodes. GPR based target helps to gain better rewards in earlier episodes itself on which the model is able to build in later episodes. This shows the robustness of GPR-MADDPG which significantly improves the performance of the learnt policy via a GPR based target for estimating better target values of the critic and limiting the fluctuations in the rewards with the help of a well-designed kernel function.TF-DQN shows the highest fluctuations in rewards depicting an unstable policy learnt using TF-DQN. Still, both TF-DQN and Meta-TD3 are able to gain higher rewards as episodes go by and are able to improve to some degree. A* based approach performs the worst in comparison to the Deep RL methods highlighting the difficulty in learning autonomous controls for UAVs in dynamic environments. The heuristic-based approach is not able to adapt to the changing environment and falls short of performing equivalent to function approximation techniques of Deep RL and also sees the highest fluctuation in rewards, depicting unstable policy.

7.3.6 Varying Number of UAVs

To explore the scalability of the proposed model, the number of UAVs in the system are varied for this experiment. All three tracks of convoy movement (straight, curved and switchback tracks) are considered for this experiment along with both variants of convoy movement behaviour. The plot in Figure 7.8(a) shows the average accumulated rewards (over 10000 episodes) with varying number of UAVs. As can be seen, for the straight track, the proposed model maintains quite a similar level of performance even with the increasing number of UAVs, however, there is a noticeable drop in the average accumulated rewards with the increase in the number of UAVs in case of curved track. A significant drop in rewards can be observed in the case of switchback track with increasing number of UAVs. This degradation in the cumulative rewards of the UAVs is due to high instability in FoV overlap due to the complexity of the curved and switchback track. The complexity of the track plays a critical role in the performance of the model, especially with the increasing number of UAVs. Still, with 14 UAVs the proposed model is able to gain positive rewards highlighting the effectiveness of the proposed model in performing cooperative tasks in complex environments with real-time tracking.

7.3.7 Impact of an RL Hyperparameter γ on Performance

In this experiment, the performance of the proposed GPR-MADDPG algorithm is observed while varying the discount factor γ. As the discount factor (γ) is one of the key RL specific hyperparameter that affects the policy learning process, sensitivity analysis for

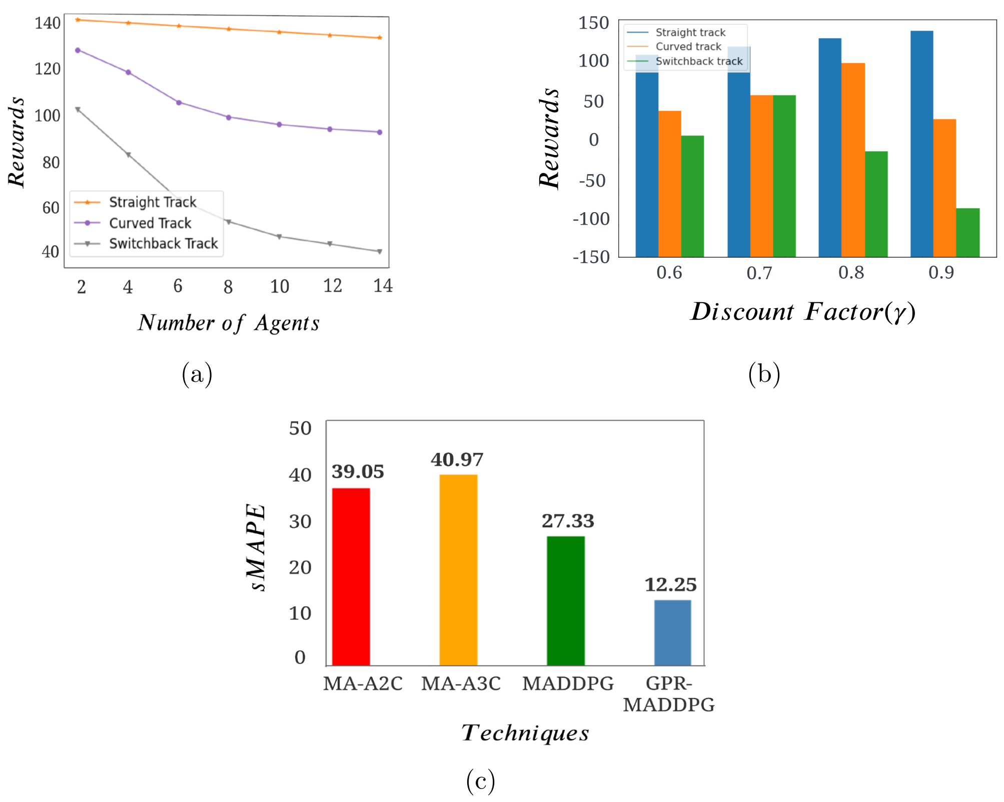


Figure 7.8: (a) Performance of GPR-MADDPG corresponding to the varying number of UAVs. (b) Sensitivity analysis for hyperparameter γ and its impact on GPR-MADDPG's performance. (c) Percentage FoV overlap among the UAVs corresponding to various methods.

γ is performed in this experiment. Figure 7.8(b) depicts the rewards accumulated by GPR-MADDPG with varying values of γ during training for all the three categories of road networks (values of γ lower than 0.6 has not been presented as the model performs sub-optimally for lower values of γ and the policy never converges). As can be seen, different values of γ have varying impact on the performance of GPR-MADDPG on different tracks. For the straight track, variation in γ's value doesn't affect the policy learning of GPR-MADDPG to a great extent. GPR-MADDPG sees almost similar performance (in terms of reward accumulation) in the case of a lower value of γ (close to 0.6) as when the value of γ is set close to 1. However, this is not true in the case of curved and switchback tracks. In the case of the curved track, the value of γ close to 0.8 is preferable, highlighting that GPR-MADDPG performance depends significantly on immediate rewards and not alone on long-term rewards. The proposed model achieves optimal learning when both immediate and long-term rewards are weighed during training. In the case of switchback track, the proposed model heavily relies on immediate rewards as it prefers the value of γ close to 0.7. This is due to the fact that in the case of switchback track there is relatively higher uncertainty in the convoy's trajectory as compared to straight and curved tracks. Value of γ around 0.8 seems to be a good trade-off for maintaining the optimal performance of the proposed approach on all different tracks.

7.3.8 Error in joint FoV Overlap

Further comparison involves examining the error in the overlap of FoVs generated by GPR-MADDPG and baseline algorithms over the considered tracks and both variants of convoy movement. An error in overlap is recorded whenever the FoVs of UAVs goes out of bound. Average symmetric Mean Absolute Percentage Error ($sMAPE$) is calculated for comparing the errors in FoVs' overlaps. $sMAPE$ for a single episode is defined as:

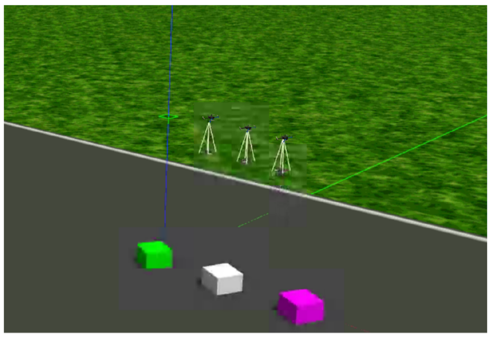
$$sMAPE = \begin{cases} \sum_{t=0}^{\Gamma} \sum_{i,j \in U, i \neq j} \frac{o_{ij}^t - 2d}{o_{ij}^t + 2d} & if\ o_{ij}^t > 2d \\ \sum_{t=0}^{\Gamma} \sum_{i,j \in U, i \neq j} \frac{\delta.2d - o_{ij}^t}{\delta.2d + o_{ij}^t} & if\ o_{ij}^t < \delta.2d \end{cases} \tag{7.18}$$

where Γ is the total time of a single episode.

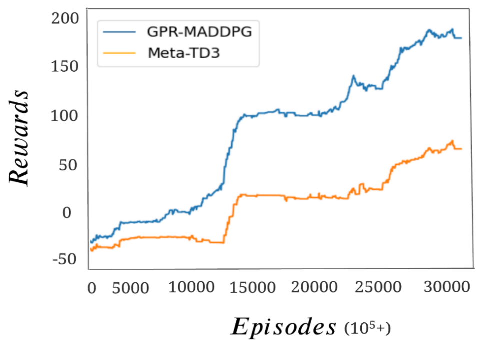
As can be observed from the plot in Figure 7.8(c), the proposed approach (GPR-MADDPG) results in the least amount of overlapping error while tracking the convoy of moving vehicles as compared to other baseline algorithms. The proposed model produces approximately 69 percent lower error as compared to MA-A2C and MA-A3C and about 55 percent lower as compared to MADDPG. This indicates that the GPR-MADDPG policy maintains a better end-to-end coverage of the moving convoy as compared to other algorithms by learning the actions that lead to better coverage and incurs minimum overlapping error.

7.3.9 Simulation in Gazebo Physics Simulator

Gazebo with Robot Operating System (ROS) provides a testbed to simulate the real-world dynamics of robotic systems. To validate the proposed model under real-world dynamics, the Gazebo 3D simulator is employed to simulate the movement of three ground robots forming a convoy, while three UAVs are deployed to track this convoy. Runway world is used for simulation where the convoy is assigned a trajectory to move along a path (with deviations and added noise) that is unknown to the UAVs.



(a)



(b)

Figure 7.9: (a) Snapshot of multiple UAVs tracking a moving convoy as implemented is Gazebo simulator. (b) Performance comparison between GPR-MADDPG and Meta-TD3 algorithms in a simulated environment in Gazebo.

Each UAV is tasked to track a single robot (see Figure 7.9(a), containing 3 UAVs and 3 robots in the Gazebo simulator). To train a GPR-MADDPG policy, a total of 130,000 episodes were conducted with random convoy trajectories in the simulator. This training was evaluated against the performance of Meta-TD3, recognized as the best algorithm in the literature. Figure 7.9(b) depicts the result of the comparison of GPR-MADDPG and Meta-TD3 in the simulator for the last 30000 episodes (as the change in accumulated rewards was very low in the initial 100000 episodes). It can be observed from the plot that the proposed GPR-MADDPG model outperforms Meta-TD3 by a large margin in simulation. A higher number of episodes were required for training in simulator as the simulated environment is very dynamic and spontaneous like the real-world environment and it's difficult for the UAVs to perform precise actions as provided by the controller. It's important to note that GPR-MADDPG operates as an off-policy method. Here, data is generated using a behavioral policy while the algorithm learns a distinct target policy. Hence, the GPR-MADDPG model can be trained using the data collected from a separate behavioural policy of UAV movement and tracking.

7.4 Chapter Summary

In this chapter a Deep RL algorithm to track a moving convoy using autonomous UAVs is proposed. The control policy is learnt for the UAVs by employing the MADDPG model, which utilizes Gaussian Process Regression (GPR) to estimate the target Q-value function. The GPR targets, coupled with an adaptive kernel function, significantly improves the final policy and expedited the training process. Comparative experiments conducted on diverse trajectories validate the robustness and superior performance of GPR-MADDPG over baseline Deep RL methods and state-of-the-art techniques in existing literature. This work can be extended to optimize UAV energy consumption and investigating the applicability of GPR targets in diverse applications. In the next chapter I conclude my thesis by summarizing the crucial findings and insights gained from the research work. It also outlines potential areas for future research, discussing avenues for further investigation.

8| Conclusions and Avenues for Future Research

In conclusion, the use of autonomous unmanned aerial vehicles (UAVs) in flood disaster relief operations holds immense potential for significantly enhancing relief planning. UAVs achieves this by reducing the requirement for additional manpower while facilitating extensive aerial coverage and offering remote sensing, real-time path planning, and various other capabilities. However, learning autonomous control policies for UAV is not trivial, especially in dynamic and stochastic environments such as floods. The contribution made in this thesis concentrate on providing Deep RL solutions for various aspects of relief operations and security applications, namely, area coverage, path planning and target tracking. The proposed Deep RL algorithms provide novel exploration strategies to improve policy learning and to expedite reward accumulation especially in early phases of training. This chapter provides a summary of the contributions made in this thesis and discusses potential directions for future research, including possible extensions to the proposed models.

8.1 Summary of the Contributions

The prime aim of this thesis is to develop and evaluate algorithms based on deep reinforcement learning (Deep RL) for autonomous multi-UAV controls tasked to operate in flood relief scenarios. These algorithms enable UAVs to effectively cover flooded areas and collect critical information within strict time-frames, attributed to limited UAV energy and the continuously changing dynamics of the flood conditions. By integrating water-flow estimation algorithms like D8 and D-infinity (DINF), this thesis addresses the challenges faced in early phases of training standard Deep RL algorithms, such as sparse rewards, random target function approximation and overestimation bias. Additionally, the research covers path planning and target tracking applications of multi-UAV systems. The algorithms proposed in this thesis provide comprehensive solutions to enable autonomous control for multi-UAV systems in various applications, including joint area coverage, real-time path planning, and moving convoy tracking.
The first contribution of this thesis introduces two new methods, D8QL and D8DQN, for multi-UAV exploration during flood disasters. These methods use the D8 flow algorithm to guide the UAVs based on water flow estimates. Compared to existing methods, D8QL and D8DQN perform better in terms of accumulated rewards and joint area coverage. However, D8QL and D8DQN have limitations. It only works with discrete action spaces and may struggle in more complex environments. Furthermore, due to the D8 algorithm, the UAVs might tend to cluster together, potentially resulting in less efficient coverage, especially when critical regions are distributed across a larger area. Building on this first

work, the second contribution of this thesis is the D3S algorithm that uses D-infinity algorithm to provide target actions with exact degree of water flow estimates to learn continuous control policies for UAVs. This technique is integrated into the DDPG model, facilitating policy learning while preventing clustering of UAVs using Path scatter. Both the proposed algorithms D8DQN and D3S highlights the significance of domain knowledge in enhancing Deep RL policies for UAVs in critical environments.

Further, recognizing the constraints posed by centrally trained algorithms like the ones mentioned above, when dealing with distributed data and communication challenges, it becomes evident that a centralized training framework might not suffice. Hence, the next contribution of this thesis explores decentralized training of UAVs to learn a local policy without relying on a ground control unit which is required in centralized training paradigms. The proposed dec-DQNC8 algorithm leverages the D8 flow estimation algorithm to enhance the exploration strategy. The algorithm enables UAVs to communicate within their transmission range, facilitating data sharing for quick and robust training. Additionally, it leverages coverage maps that contains the observed trajectories of each individual UAV. This information enhances the UAVs' awareness of each other's paths and helps in maximizing the overall area coverage. Results from training and test environments showcase the algorithm's significant impact on realizing successful decentralized multi-UAV policies, demonstrating improvements across multiple performance metrics.

Expanding the application scope, the subsequent application involved addressing the problem of path planning to reach critical locations during floods. This contribution presents an automated real-time path-planning algorithm for UAVs to assist waterborne evacuation vehicles (WBVs) to reach critical location(s) during floods. The UAVs maintain an intervened formation to sense connected regions that are serviceable i.e., devoid of obstacles and shallow water regions, for possible movement of WBVs. Initial exploration is enabled along a guide path i.e., the shortest path from the location of the WBV to the critical location, due to limited battery of UAVs. The neighbouring regions are sensed by the UAVs if the guide path is found to be unserviceable. The proposed algorithm MEA*MADDPG is compared with existing path-planning techniques, showcasing its effectiveness in generating real-time serviceable paths.

Furthermore, the application of tracking a moving convoy of vehicles using autonomous UAVs is also addressed in this thesis. A Deep RL algorithm is proposed that uses Gaussian Process Regression (GPR) to approximate the target Q-value function for MADDPG model. Utilizing GPR targets along with an adaptive kernel function allowed for a more precise approximation. Experimentation using various road trajectories of differing complexities demonstrated the robustness and superior performance of GPR-MADDPG over baseline Deep RL approaches, as well as state-of-the-art techniques from existing literature.

All the proposed algorithm in this thesis offer effective autonomous solutions for multi-UAV systems across various applications, providing enhanced coverage, real-time path planning, and consistent target tracking.

8.2 Future Research Scopes

The research work reported in the contributing chapters of this thesis provides ample scope to advance the work in various research directions. As part of future work, investigating the efficacy of D8-enhanced Double DQN (D8-DDQN) and Dueling DQN can offer deeper insights into the performance of D8-based RL models (Chapter 3). Additionally, extending non-RL techniques like Genetic Algorithms (GA), A*, etc., to learn continuous UAV actions and comparing them with deep RL models could provide valuable insights (Chapter 4). Moreover, analyzing the impact of varying number of UAVs on overall coverage and rewards can help fine-tune incentive and penalty configurations (Chapter 4). In reference to the proposed model, dec-DQNC8, extending decentralized training to continuous action spaces with Deep RL models like DDPG, SAC, and TD3 for multi-UAV systems holds promise for improving scalability and efficiency (Chapter 5). Further, exploring variations in communication settings such as transmission range, power, or probability can offer valuable insights, contributing to the enhancement of dec-DQNC8's performance (Chapter 5). Additionally, investigating obstacle-bound environments such as tunnels can be interesting for testing the performance of GPR-MADDPG based multi-UAV convoy tracking, where UAVs encounter unique challenges (Chapter 7).

Future work may also involve the development of energy-aware multi-UAV systems, including techniques for autonomous landing on moving platforms to recharge, thus enhancing operational endurance. In addition, exploring data-driven approaches to dynamically adjust kernel parameters based on observed interactions could enhance algorithm adaptability (Chapter 7). Techniques like SHAP (SHapley Additive exPlanations) can be investigated for interpretable policy actions. One of the primary domains of future work is to extend the proposed solutions in various other disaster scenarios, such as earthquakes, avalanches, wildfires etc. By carefully attributing the domain characteristics, better target functions can be designed that are closer to true value functions. Additionally, exploring transfer learning to initialize network weights can be valuable when employing a trained model from a disaster scenario to a similar application. Considering the proposed algorithms, extending the action set to integrate altitude control could significantly enhance adaptability, particularly in tracking applications where the moving objects might exit the field-of-view. In such instances, UAVs require the capability to readjust and relocate the objects, and by incorporating altitude control, the UAVs can efficiently handle these situations, ensuring continuous tracking.

Additionally, exploring energy modeling for UAVs beyond flight time limitations involves investigating strategies for optimizing UAV energy consumption. This extension can prolong the UAVs' flight duration, thus extending their support capabilities during disaster response operations. Further exploration into energy modeling, maintaining optimal altitude, and utilizing trajectory optimization techniques can further extend the flight time of UAVs. Moreover, investigating reward shaping to enhance multi-UAV policies can be a potential avenue for future research in disaster relief applications. Reward shaping involves customizing a reward function to provide more frequent feedback on desired behaviors, an area currently drawing attention from researchers in the field of reinforcement learning. Timely feedback proves crucial, especially during initial learning phases, as it encourages the exploration of favorable behaviors from the outset. Incorporating diverse methodologies like domain knowledge based exploration, energy optimization and reward shaping, exhibits promising potential to extend the impact of Deep RL algorithms across diverse applications. These methods hold the promise of significantly enhancing operational strategies, offering avenues for continual improvement and innovation in addressing challenges within dynamic and stochastic environments, such as disaster response scenarios.

9| Dataset & Code

- **D8QL and D8DQN**
  Link: https://github.com/Armaan-Garg/D8QL-D8DQN

- **D3S**
  Link: https://github.com/Armaan-Garg/D3S

- **dec-DQNC8**
  Link: https://github.com/Armaan-Garg/dec-DQNC8

- **MEA*MADDPG**
  Link: https://github.com/Armaan-Garg/MEA-MADDPG

- **GPR-MADDPG**
  Dataset: https://github.com/Armaan-Garg/Dataset-GPR-MADDPG
  Link: https://github.com/Armaan-Garg/GPR-MADDPG

References

[1] Chennai flood map. URL http://sreejithr.in/chennai-floods/.

[2] UNDRR. Human cost of disasters, an overview of the last 20 years, 2000-2019. URL https://www.undrr.org/media/48008/download?startDownload=true.

[3] Hannah Ritchie, Pablo Rosado, and Max Roser. Natural disasters, Dec 2022. URL https://ourworldindata.org/natural-disasters.

[4] Mohamed Abdelkader, Mohammad Shaqura, Christian G. Claudel, and Wail Gueaieb. A uav based system for real time flash flood monitoring in desert environments using lagrangian microsensors. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 25–34, 2013.

[5] Alex Wallar, Erion Plaku, and Donald A. Sofge. Reactive motion planning for unmanned aerial surveillance of risk-sensitive areas. *IEEE Transactions on Automation Science and Engineering*, 12(3):969–980, 2015.

[6] Hafiz Suliman Munawar, Ahmed W.A. Hammad, and S. Travis Waller. Disaster region coverage using drones: Maximum area coverage and minimum resource utilisation. *Drones*, 6(4), 2022. ISSN 2504-446X.

[7] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4): 837–855, 2018.

[8] Zainal Mahayuddin, Hairina Jais, and Haslina Arshad. Comparison of human pilot (remote) control systems in multirotor unmanned aerial vehicle navigation. *International Journal on Advanced Science, Engineering and Information Technology*, 7:132, 02 2017.

[9] Asif Ali Laghari, Awais Khan Jumani, Rashid Ali Laghari, and Haque Nawaz. Unmanned aerial vehicles: A review. *Cognitive Robotics*, 3:8–22, 2023. ISSN 2667-2413.

[10] Daniele Giordan, Marc Adams, Irene Aicardi, Maria Alicandro, Paolo Allasia, Marco Baldo, Pierluigi Berardinis, Donatella Dominici, Danilo Godone, Peter Hobbs, Veronika Lechner, Tomasz Niedzielski, Marco Piras, Marianna Rotilio, Riccardo Salvini, Valerio Segor, Bernadette Sotier, and Fabrizio Troilo. The use of unmanned aerial vehicles (uavs) for engineering geology applications. *Bulletin of Engineering Geology and the Environment*, 79, 04 2020.

[11] Eivind Brastad Dammen. Reinforcement learning and evolutionary algorithms for attitude control, a comparison for aerial vehicles, 2022. URL https://www.duo.uio.no/handle/10852/95569.

[12] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020. ISSN 0306-2619.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[14] Ahmad Taher Azar, Anis Koubaa, Nada Ali Mohamed, Habiba A. Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa M. Khamis, Ibrahim A. Hameed, and Gabriella Casalino. Drone deep reinforcement learning: A review. *Electronics*, 10(9):999, 2021. ISSN 2079-9292.

[15] Fadi AlMahamid and Katarina Grolinger. Autonomous unmanned aerial vehicle navigation using reinforcement learning: A systematic review. *Engineering Applications of Artificial Intelligence*, 115:105321, 2022. ISSN 0952-1976.

[16] Tiberius-Florian Frigioescu, Mihaela Raluca Condruz, Teodor Adrian Badea, and Alexandru Paraschiv. A preliminary study on the development of a new uav concept and the associated flight method. *Drones*, 7(3):166, 2023.

[17] Georgy Skorobogatov, Cristina Barrado, and Esther Salamí. Multiple uav systems: A survey. *Unmanned Systems*, 08(02):149–169, 2020.

[18] M. L. Cummings. Operator interaction with centralized versus decentralized uav architectures. *Handbook of Unmanned Aerial Vehicles*, page 977–992, 2014.

[19] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. The MIT Press, 2018.

[20] Aske Plaat, Walter Kosters, and Mike Preuss. High-accuracy model-based reinforcement learning, a survey. *Artificial Intelligence Review*, 2023.

[21] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric D. Langlois, Matthew Shunshi Zhang, Guodong Zhang, P. Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. 2019. URL `http://arxiv.org/abs/1907.02057`.

[22] Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning, 2017. URL `http://arxiv.org/abs/1707.06203`.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL http://arxiv.org/abs/1712.01815.

[24] Phillip Swazinna, Steffen Udluft, Daniel Hein, and Thomas Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning. *IFAC-PapersOnLine*, 55(15):19–26, 2022. ISSN 2405-8963. 6th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2022.

[25] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, page 2094–2100, 2016.

[26] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1995–2003, 2016.

[27] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[28] David Baldazo, Juan Parras, and Santiago Zazo. Decentralized multi-agent deep reinforcement learning in swarms of drones for flood monitoring. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[29] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. Uav coverage path planning under varying power constraints using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1444–1449, 2020.

[30] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, page 1928–1937, 2016.

[31] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 5285–5294, 2017. ISBN 9781510860964.

[32] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865, 2018.

[34] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018.

[35] Qiming Yang, Yan Zhu, Jiandong Zhang, Shasha Qiao, and Jieling Liu. Uav air combat autonomous maneuver decision based on ddpg algorithm. In *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pages 37–42, 2019.

[36] Senyan Zhao, Wei Wang, Jun Li, Subin Huang, and Sanmin Liu. Autonomous navigation of the uav through deep reinforcement learning with sensor perception enhancement. *Mathematical Problems in Engineering*, 2023:1–11, 2023.

[37] Armaan Garg and Shashi Shekhar Jha. Deep deterministic policy gradient based multi-uav control for moving convoy tracking. *Engineering Applications of Artificial Intelligence*, 126:107099, 2023. ISSN 0952-1976.

[38] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(1):237–285, 1996. ISSN 1076-9757.

[39] Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. The problem with ddpg: understanding failures in deterministic environments with sparse rewards. In *International Conference on Artificial Neural Networks*, 2019.

[40] Seungchan Kim, Kavosh Asadi, Michael L. Littman, and George Dimitri Konidaris. Deepmellow: Removing the need for a target network in deep q-learning. In *International Joint Conference on Artificial Intelligence*, 2019.

[41] Andrew Silva and Matthew Gombolay. Encoding human domain knowledge to warm start reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5042–5050, 2021.

[42] Rajkumar Ramamurthy, Christian Bauckhage, Rafet Sifa, Jannis Schücker, and Stefan Wrobel. Leveraging domain knowledge for reinforcement learning using mmc architectures. *Lecture Notes in Computer Science*, page 595–607, 2019.

[43] Yuedong Wang, Yan Liang, Huixia Zhang, and Yijing Gu. Domain knowledge-assisted deep reinforcement learning power allocation for mimo radar detection. *IEEE Sensors Journal*, 22(23):23117–23128, 2022.

[44] Armaan Garg and Shashi Shekhar Jha. Directed explorations during flood disasters using multi-uav system. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 2154–2161, 2022.

[45] Pin-Chun Huang. Analysis of hydrograph shape affected by flow-direction assumptions in rainfall-runoff models. *Water*, 12(2), 2020. ISSN 2073-4441.

[46] Petter Pilesjö and Abdulghani Hasan. A triangular form-based multiple flow algorithm to estimate overland flow distribution and accumulation on a digital elevation model. *Transactions in GIS*, 18(1):108–124, 2013.

[47] Prakarsh Kaushik, Armaan Garg, and Shashi Shekhar Jha. On learning multi-uav policy for multi-object tracking and formation control. In *2021 IEEE 18th India Council International Conference (INDICON)*, pages 1–6, 2021.

[48] Guillaume Bono, Jilles Steeve Dibangoye, Laëtitia Matignon, Florian Pereyron, and Olivier Simonin. Cooperative multi-agent policy gradient. *Machine Learning and Knowledge Discovery in Databases*, page 459–476, 2019.

[49] Raúl Arranz, David Carramiñana, Gonzalo de Miguel, Juan A. Besada, and Ana M. Bernardos. Application of deep reinforcement learning to uav swarming for ground surveillance. *Sensors*, 23(21):8766, 2023.

[50] Mikko Lauri, Joni Pajarinen, and Jan Peters. Information gathering in decentralized pomdps by policy graph improvement. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS ’19, page 1143–1151. International Foundation for Autonomous Agents and Multiagent Systems, 2019. ISBN 9781450363099.

[51] Chongjie Zhang and Victor Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS ’13, page 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems, 2013. ISBN 9781450319935.

[52] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *AAMAS ’21: 20th International Conference on Autonomous Agents and Multiagent*

Systems, Virtual Event, United Kingdom, May 3-7, 2021, pages 844–852. ACM, 2021.

[53] Yongheng Liang, Hejun Wu, and Haitao Wang. Asm-ppo: Asynchronous and scalable multi-agent ppo for cooperative charging. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 798–806. International Foundation for Autonomous Agents and Multiagent Systems, 2022. ISBN 9781450392136.

[54] Osrm api documentation. URL https://project-osrm.org/docs/v5.24.0/api/#.

[55] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[56] Yeonsoo Kim and Jong Min Lee. Model-based reinforcement learning for nonlinear optimal control with practical asymptotic stability guarantees. *AIChE Journal*, 66 (10), 2020.

[57] Runlong Miao, Lingxiao Wang, and Shuo Pang. Coordination of distributed unmanned surface vehicles via model-based reinforcement learning methods. *Applied Ocean Research*, 122:103106, 2022.

[58] David A. Mottice. Team air combat using model-based reinforcement learning, 2022. URL https://scholar.afit.edu/etd/5364/.

[59] Christopher J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4): 279–292, 1992.

[60] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[61] Shiqi Zhang and Mohan Sridharan. A survey of knowledge-based sequential decision-making under uncertainty. *AI Magazine*, 43(2):249–266, 2022.

[62] Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Aria Nefian. Cooperative and distributed reinforcement learning of drones for field coverage, 2018. URL http://arxiv.org/abs/1803.07250.

[63] Bo Liu, Yue Zhang, Shupo Fu, and Xuan Liu. Reduce uav coverage energy consumption through actor-critic algorithm. In *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 332–337, 2019.

[64] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen. Reinforcement learning for autonomous uav navigation using function approximation. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2018.

[65] Qianqian Wu, Qiang Liu, Zefan Wu, and Jiye Zhang. Maximizing uav coverage in maritime wireless networks: A multiagent reinforcement learning approach. *Future Internet*, 15(11):369, 2023.

[66] Kingsley Nweye, Bo Liu, Peter Stone, and Zoltan Nagy. Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings. *Energy and AI*, 10: 100202, 2022.

[67] Armaan Garg and Shashi Shekhar Jha. Decentralized critical area coverage using multi-uav system with guided explorations during floods. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6, 2023.

[68] Christian Zammit and Erik-Jan Van Kampen. Comparison between a* and rrt algorithms for uav path planning. *2018 AIAA Guidance, Navigation, and Control Conference*, 2018.

[69] Ghulam Farid, Silvio Cocuzza, Talha Younas, Asghar Abbas Razzaqi, Waqas Ahmad Wattoo, Ferdinando Cannella, and Hongwei Mo. Modified a-star (a*) approach to plan the motion of a quadrotor uav in three-dimensional obstacle-cluttered environment. *Applied Sciences*, 12(12):5791, 2022.

[70] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. *Journal of Intelligent amp; Robotic Systems*, 98(2):297–309, 2019.

[71] Alejandro Puente-Castro, Daniel Rivero, Eurico Pedrosa, Artur Pereira, Nuno Lau, and Enrique Fernandez-Blanco. Q-learning based system for path planning with unmanned aerial vehicles swarms in obstacle environments. *Expert Systems with Applications*, 235:121240, 2023. ISSN 0957-4174.

[72] Nouman Bashir, Saadi Boudjit, Gabriel Dauphin, and Sherali Zeadally. An obstacle avoidance approach for uav path planning. *Simulation Modelling Practice and Theory*, 129:102815, 2023. ISSN 1569-190X.

[73] Armaan Garg and Shashi Shekhar Jha. Real-time serviceable path planning using uavs for waterborne vehicle navigation during floods. In *Proceedings of Conference on Advances In Robotics*. Association for Computing Machinery, 2023.

[74] Alejandro Puente-Castro, Daniel Rivero, Alejandro Pazos, and Enrique Fernandez-Blanco. Uav swarm path planning with reinforcement learning for field prospecting. *Applied Intelligence*, 2022.

[75] Jiawei Liang. Vision-based unmanned aerial vehicle navigation in virtual complex environment using deep reinforcement learning, Feb 2022. URL https://escholarship.org/uc/item/6zb8r46w.

[76] Yintao Zhang, Youmin Zhang, and Ziquan Yu. Path following control for uav using deep reinforcement learning approach. *Guidance, Navigation and Control*, 01(01): 2150005, 2021.

[77] Wenhong ZHOU, Jie LI, Zhihong LIU, and Lincheng SHEN. Improving multi-target cooperative tracking guidance for uav swarms using multi-agent reinforcement learning. *Chinese Journal of Aeronautics*, 35(7):100–112, 2022.

[78] Armaan Garg and Shashi Shekhar Jha. Deep deterministic policy gradient based multi-uav control for moving convoy tracking. *Engineering Applications of Artificial Intelligence*, 126:107099, 2023. ISSN 0952-1976.

[79] Bo Li, Zhigang Gan, Daqing Chen, and Dyachenko Sergey Aleksandrovich. Uav maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning. *Remote Sensing*, 12(22):3789, 2020.

[80] Sarthak Bhagat and P.B. Sujit. Uav target tracking in urban environments using deep reinforcement learning. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020.

[81] Nicola Lissandrini, Giulia Michieletto, Riccardo Antonello, Marta Galvan, Alberto Franco, and Angelo Cenedese. Cooperative optimization of uavs formation visual tracking. *Robotics*, 8(3):52, 2019.

[82] D. Bianchi, A. Borri, S. Di Gennaro, and M. Preziuso. Uav trajectory control with rule-based minimum-energy reference generation. In *2022 European Control Conference (ECC)*, pages 1497–1502, 2022.

[83] Markus Ortlieb and Florian-Michael Adolf. Rule-based path planning for unmanned aerial vehicles in non-segregated air space over congested areas. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–9, 2020.

[84] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and ai. *Robotics and Autonomous Systems*, 154:104096, 2022.

[85] Qian Huang, Xianming Ma, Kun Liu, Xinyi Ma, and Weijian Pang. Autonomous reconnaissance action of swarm unmanned system driven by behavior tree. In *2022 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1540–1544, 2022.

[86] Ankit Agrawal, Sophia J. Abraham, Benjamin Burger, Chichi Christine, Luke Fraser, John M. Hoeksema, Sarah Hwang, Elizabeth Travnik, Shreya Kumar, Walter

Scheirer, Jane Cleland-Huang, Michael Vierhauser, Ryan Bauer, and Steve Cox. The next generation of human-drone partnerships: Co-designing an emergency response system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080.

[87] Yue Fan, Shilei Chu, Wei Zhang, Ran Song, and Yibin Li. Learn by observation: Imitation learning for drone patrolling from videos of a human navigator. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5209–5216, 2020.

[88] Mohamed Rabah, Ali Rohan, Sherif A.S. Mohamed, and Sung Kim. Autonomous moving target-tracking for a uav quadcopter based on fuzzy-pi. *IEEE Access*, 7: 38407 – 38419, 03 2019.

[89] Abel Hailemichael and Ali Karimoddini. Development of a robust interval type-2 tsk fuzzy logic controlled uav platform. *Journal of Intelligent amp; Robotic Systems*, 107(2), 2023.

[90] Jibin Zheng, Minghui Ding, Lu Sun, and Hongwei Liu. Distributed stochastic algorithm based on enhanced genetic algorithm for path planning of multi-uav cooperative area search. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2023.

[91] Changhee Han. Applying ga as autonomous landing methodology to a computer-simulated uav. *Computational Science/Intelligence and Applied Informatics*, page 49–63, 2019.

[92] Xiaobing Yu, Chenliang Li, and JiaFang Zhou. A constrained differential evolution algorithm to solve uav path planning in disaster scenarios. *Knowledge-Based Systems*, 204:106209, 2020.

[93] Hazha Saeed Yahia and Amin Salih Mohammed. Path planning optimization in unmanned aerial vehicles using meta-heuristic algorithms: A systematic review. *Environmental Monitoring and Assessment*, 195(1), 2022.

[94] Haolong Zheng. Ant colony optimization based uav path planning for autonomous agricultural spraying. In *2022 IEEE 5th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pages 910–916, 2022.

[95] Sheng Gao, Jiazheng Wu, and Jianliang Ai. Multi-uav reconnaissance task allocation for heterogeneous targets using grouping ant colony optimization algorithm. *Soft Computing*, 25(10):7155–7167, 2021.

[96] Mohammed A. Alanezi, Houssem R. E. H. Bouchekara, Tijani Abdul-Aziz Apalara, Mohammad Shoaib Shahriar, Yusuf A. Sha'aban, Muhammad Sharjeel Javaid,

and Mohammed Abdallah Khodja. Dynamic target search using multi-uavs based on motion-encoded genetic algorithm with multiple parents. *IEEE Access*, 10: 77922–77939, 2022.

[97] Zakria Qadir, Muhammad Hamza Zafar, Syed Kumayl Moosavi, Khoa N. Le, and Vivian W. Tam. Optimizing uav path for disaster management in smart cities using metaheuristic algorithms. *Studies in Computational Intelligence*, page 225–244, 2022.

[98] Yaohong Qu, Ying Sun, Kai Wang, and Feng Zhang. Multi-uav cooperative search method for a moving target on the ground or sea. In *2019 Chinese Control Conference (CCC)*, pages 4049–4054, 2019.

[99] Li Yan, Mohd Wazih Ahmad, Malik Jawarneh, Mohammad Shabaz, R. Raffik, and Kakarla Hari Kishore. Single-input single-output system with multiple time delay pid control methods for uav cluster multiagent systems. *Security and Communication Networks*, 2022:1–7, 2022.

[100] Luca Cavanini, Gianluca Ippoliti, and Eduardo F. Camacho. Model predictive control for a linear parameter varying model of an uav. *Journal of Intelligent amp; Robotic Systems*, 101(3), 2021.

[101] Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4): 2114–2130, 2022.

[102] Animesh Sahu, Harikumar Kandath, and K. Madhava Krishna. Model predictive control based algorithm for multi-target tracking using a swarm of fixed wing uavs. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1255–1260, 2021.

[103] Ningjun LIU, Zhihao CAI, Jiang ZHAO, and Yingxun WANG. Predictor-based model reference adaptive roll and yaw control of a quad-tiltrotor uav. *Chinese Journal of Aeronautics*, 33(1):282–295, 2020.

[104] Ziyang Zhen, Gang Tao, Yue Xu, and Ge Song. Multivariable adaptive control based consensus flight control system for uavs formation. *Aerospace Science and Technology*, 93:105336, 2019.

[105] Christos E. Papoutsellis. Numerical simulation of non-linear water waves over variable bathymetry. *Procedia Computer Science*, 66:174–183, 2015. ISSN 1877-0509. 4th International Young Scientist Conference on Computational Science.

[106] Pin-Chun Huang and Kwan Tun Lee. An efficient method for dem-based overland flow routing. *Journal of Hydrology*, 489:238–245, 2013. ISSN 0022-1694.

[107] M. Rajeevan, Jyoti Bhate, and A. K. Jaswal. Analysis of variability and trends of extreme rainfall events over india using 104 years of gridded daily rainfall data. *Geophysical Research Letters*, 35(18), 2008.

[108] William S. Gonwa and M. Levent Kavvas. A modified diffusion equation for flood propagation in trapezoidal channels. *Journal of Hydrology*, 83(1):119–136, January 1986.

[109] Mohamed-Ayoub Messous, Hichem Sedjelmaci, and Sidi-Mohammed Senouci. Implementing an emerging mobility model for a fleet of uavs based on a fuzzy logic inference system. *Pervasive and Mobile Computing*, 42:393–410, 2017. ISSN 1574-1192.

[110] Rutuja Shivgan and Ziqian Dong. Energy-efficient drone coverage path planning using genetic algorithm. In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, 2020.

[111] Hoang Viet, Viet-Hung Dang, SeungYoon Choi, and Tae Chung. Bob: an online coverage approach for multi-robot systems. *Applied Intelligence*, 42, 03 2014.

[112] Mapbox bathymetry. URL https://docs.mapbox.com/android/legacy/maps/examples/bathymetry/. Accessed: 2022-06-13.

[113] Flood mapping tool user guide. URL https://floodmapping.inweh.unu.edu/. Accessed: 2022-06-13.

[114] Mapbox terrain-dem v1: Tilesets. URL https://docs.mapbox.com/data/tilesets/reference/mapbox-terrain-dem-v1/. Accessed: 2022-06-13.

[115] David G. Tarboton. A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water Resources Research*, 33(2):309–319, 1997.

[116] Chris Veness. Calculate distance and bearing between two latitude/longitude points using haversine formula. URL https://www.movable-type.co.uk/scripts/latlong.html.

[117] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance. *IEEE Access*, 6:58383–58394, 2018.

[118] Guillaume Matheron, Nicolas Perrin 0001, and Olivier Sigaud. Understanding failures of deterministic actor-critic with continuous action spaces and sparse rewards. In *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia,*

September 15-18, 2020, Proceedings, Part II, volume 12397, pages 308–320, 2020. ISBN 978-3-030-61616-8.

[119] Archana Patankar. The exposure, vulnerability, and ability to respond of poor households to recurrent floods in mumbai. *Policy Research Working Papers*, 2015.

[120] Engie Bashir and Mitja Luštrek. *Intelligent Environments 2021: Workshop proceedings of the 17th International Conference on Intelligent Environments*. IOS Press, 2021.

[121] Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. Random walk exploration for swarm mapping. In *Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019, London, UK, July 3–5, 2019, Proceedings, Part II*, page 211–222, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-25331-8.

[122] Shouqi Wang, Yongqiang Bai, and Chao Zhou. Coverage path planning design of mapping uavs based on particle swarm optimization algorithm. In *2019 Chinese Control Conference (CCC)*, pages 8236–8241, 2019.

[123] Barna Róbert, Kristóf Solymosi, and Eleonóra Stettner. Mathematical analysis of drone flight path. *Journal of Agricultural Informatics*, 10, 12 2019.

[124] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. *Autonomous Agents and Multiagent Systems*, page 66–83, 2017.

[125] Yong Zeng, Jie Xu, and Rui Zhang. Energy minimization for wireless communication with rotary-wing uav. *IEEE Transactions on Wireless Communications*, 18(4): 2329–2345, 2019.

[126] Ran Zhuo, Shiqian Song, and Yejun Xu. Uav communication network modeling and energy consumption optimization based on routing algorithm. *Computational and Mathematical Methods in Medicine*, 2022:1–10, 2022.

[127] Roger McFarlane. A survey of exploration strategies in reinforcement learning, 2003. URL https://www.cs.mcgill.ca/~cs526/roger.pdf.

[128] Hamed Hellaoui, Ali Chelli, Miloud Bagaa, and Tarik Taleb. Uav communication strategies in the next generation of mobile networks. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1642–1647, 2020.

[129] What is snr?, 2015. URL https://mason.gmu.edu/~rmorika2/What_is_SNR_.htm. Accessed: 2022-06-13.

[130] Hendrik Lumbantoruan and Koichi Adachi. Array antenna equipped uav-bs for efficient low power wsn and its theoretical analysis. *IET Communications*, 15(16): 2054–2067, 2021.

[131] Tharindu D. Ponnimbaduge Perera, Dushantha Nalin K. Jayakody, Sahil Garg, Neeraj Kumar, and Ling Cheng. Wireless- powered uav assisted communication system in nakagami-m fading channels. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2020.

[132] Jesús Sánchez-García, Daniel Gutiérrez, and S.L. Toral. A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario. *Future Generation Computer Systems*, 90, 07 2018.

[133] Mapbox streets, 2023. URL https://docs.mapbox.com/data/tilesets/reference/mapbox-streets-v8/. Accessed: 2023-1-1.

[134] João Braun, Thadeu Brito, José Lima, Paulo Costa, Pedro Costa, and Alberto Nakano. A comparison of a* and rrt* algorithms with dynamic and real time constraint scenarios for mobile robots. In *Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, page 398–405, 2019. ISBN 9789897583810.

[135] Ariel Felner, Meir Goldenberg, Guni Sharon, Roni Stern, Tal Beja, Nathan Sturtevant, Jonathan Schaeffer, and Robert C. Holte. Partial-expansion a* with selective node generation. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 471–477. AAAI Press, 2012.

[136] Huy X. Pham, Hung M. La, David Feil-Seifer, and Matthew Deans. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6648–6653, 2017. doi: 10.1109/IROS.2017.8206579.

[137] Jauwairia Nasir, Fahad Islam, Usman Ali Malik, Yasar Ayaz, Osman Hasan, Mushtaq Khan, and Mannan Muhammad. Rrt*-smart: A rapid convergence implementation of rrt*,. *International Journal of Advanced Robotic Systems*, 10: 299, 07 2013. doi: 10.5772/56718.

[138] Weixin Yang, Gang Wang, and Yantao Shen. Perception-aware path finding and following of snake robot in unknown environment. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5925–5930, 2020.

[139] Hao Liu and P. Abbeel. Behavior from the void: Unsupervised active pre-training. In *Neural Information Processing Systems*, 2021.

[140] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, page 514–520, Cambridge, MA, USA, 1995. MIT Press.

[141] Stefanos Leonardos and Georgios Piliouras. Exploration-exploitation in multi-agent learning: Catastrophe theory meets game theory. *Artificial Intelligence*, 304:103653, 2022. ISSN 0004-3702.

[142] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. *Proceedings of the 22nd international conference on Machine learning - ICML 05*, 2005.

[143] Marc Deisenroth, Carl Rasmussen, and Jan Peters. Gaussian process dynamic programming. *Neurocomputing*, 72:1508–1524, 03 2009.

[144] Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[145] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS Datasets and Benchmarks*, 2020.

[146] Saeid Nahavandi Ngoc Duy Nguyen, Thanh Thi Nguyen. A visual communication map for multi-agent deep reinforcement learning, 2020. URL https://arxiv.org/abs/2002.11882.

[147] Yingzhe Cai, Qingbiao Xi, Xiaojun Xing, Haoran Gui, and Qi Liu. Path planning for uav tracking target based on improved a-star algorithm. In *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, pages 1–6, 2019.

[148] Tommaso Villa, Felipe Gonzalez, Branka Miljievic, Zoran Ristovski, and Lidia Morawska. An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives. *Sensors*, 16(7):1072, 2016.