Multi-agent Q-learning with particle filtering for UAV tracking in Open-RAN environment

Soleymani, Seyed Ahmad; Goudarzi, Shidrokh; Xiao, Pei; Mihaylova, Lyudmila; Shojafar, Mohammad; Wang, Wenwu

Soleymani, S. A., Goudarzi, S., Xiao, P., Mihaylova, L., Shojafar, M., & Wang, W. (2025). Multi-agent Q-learning with particle filtering for UAV tracking in Open-RAN environment. IEEE Transactions on Aerospace and Electronic Systems, 61(4), 10439–10458. https://doi.org/10.1109/TAES.2025.3559518
Document Version: Author's Accepted Manuscript

Published Version: https://doi.org/10.1109/TAES.2025.3559518

Multi-agent Q-learning with Particle Filtering for UAV Tracking in Open-RAN Environment

Seyed Ahmad Soleymani, Fellow, IEEE
University of Surrey, Guildford, UK.

Shidrokh Goudarzi, Member, IEEE
University of West London, London, UK.

Pei Xiao,, Senior Member, IEEE
University of Surrey, Guildford, UK.

Lyudmila Mihaylova, Senior Member, IEEE
University of Sheffield, UK.

Mohammad Shojafar, Senior Member, IEEE
University of Surrey, Guildford, UK.

Wenwu Wang, Senior Member, IEEE
University of Surrey, Guildford, UK.

***Abstract*—This paper introduces a method for target tracking that leverages mobile sensor nodes and Unmanned Aerial Vehicles (UAVs) within an Open- Radio Access Network (RAN) framework. Open-RAN is a flexible and standardized architecture that allows open and interoperable components in RANs, promoting efficiency and adaptability. The core methodology involves improving the accuracy and energy consumption tracking in urban areas filled with obstacles and dynamic conditions. Mobile sensor nodes use a particle filtering algorithm to detect and estimate target positions, and this information is relayed to nearby Evolved/Next Generation Node Bs (e/gNBs), which function as the radio access network infrastructure. The e/gNBs manage clusters of UAVs using a specialized xApp integrated into the near-real-time RAN Intelligent Controller (RIC). The UAVs utilize a comprehensive tracking strategy based on received signal strength (RSS), a trilateration algorithm, and an enhanced multi agent Q-learning algorithm (eMAQL). This approach enables UAVs to optimize their flight paths while balancing accuracy, power usage, and communication delays. The experimental results show that the system achieves optimal performance with eight discrete actions for eMAQL, with UAVs consuming an average of 90 ($watts$) of power and maintaining a root mean square error (RMSE) of less than 0.5 ($meters$) for target position estimation. These results highlight the system's effectiveness in providing precise and energy-efficient tracking in complex urban environments.**

***Index Terms*—Target Tracking, UAV, Q-Learning, Particle Filtering, RSS, Open-RAN, Accuracy, Power Consumption, Delay.**

S.A. Soleymani, P. Xiao, M. Shojafar are with the Institute for Communication Systems (5GIC), University of Surrey, Guildford, GU2 7XH, UK. (e-mail: s.soleymani@surrey.ac.uk;p.xiao@surrey.ac.uk;m.shojafar@surrey.ac.uk). S. Goudarzi is with the School of Computing and Engineering, University of West London, London W5 5RF, UK. (e-mail: shidrokh.goudarzi@uwl.ac.uk). L. Mihaylova is with the Department of Automatic Control and Systems Engineering, The University of Sheffield, UK. (e-mail: l.s.mihaylova@sheffield.ac.uk). W. Wang is with the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, GU2 7XH, UK. (e-mail: w.wang@surrey.ac.uk).

I. INTRODUCTION

Cooperative sensing is a collaborative approach in which multiple sensing entities, such as UAVs, work together to achieve a common sensing goal by sharing resources, information and processing capabilities. This approach improves accuracy, coverage, and reliability compared to individual sensors operating in isolation. In UAV networks, cooperative sensing has emerged as a promising technique to gather and share information, enabling improved performance in applications such as surveillance, environmental monitoring, and object detection [1]. UAVs have revolutionized various industries by offering versatile and cost-effective solutions, particularly in tracking targets for ground-based environments. Their ability to autonomously monitor above targets improves situational awareness and enables efficient decision making in domains such as search and rescue, infrastructure inspection, and security operations [2].

UAV-based target tracking involves the utilization of UAVs equipped with camera, radio frequency (RF) sensors, and advanced algorithms, to detect, locate, and track moving targets. In this setup, the camera component is leveraged for target detection and tracking purposes, while RF sensors are used for navigation and localization tasks [3]. By combining the capabilities of these sensor types, UAVs can effectively monitor and pursue dynamic targets, enabling a comprehensive and versatile approach to target tracking in diverse scenarios. Using their aerial perspective, UAVs can overcome many limitations of traditional ground-based tracking systems, such as line-of-sight obstructions and limited coverage.

However, despite the notable progress achieved in UAV-based target tracking, there remain several persistent challenges that have impeded the development of robust and efficient tracking systems, particularly in ground-based environments. These challenges are a result of both the limitations of UAVs and the inherent complexities involved in tracking dynamic targets on the ground, including vehicles and pedestrians.

In challenging urban environments, the performance of UAV-based target tracking systems can be significantly compromised due to the presence of obstacles that obstruct the signal reception by the UAVs. This obstacle-induced signal loss adversely affects the tracking process, leading to missed detection and inaccuracy, thereby reducing the overall effectiveness of the system [4], [5]. This challenge becomes more highlighted when employing location estimation techniques such as time-

of-arrival (TOA), angle-of-arrival (AOA), and received signal strength (RSS) in target tracking systems. As explained in [6], the propagation of the signal involves two primary factors: path loss model parameters and signal transmission power. Obtaining access to these parameters in challenging and unfamiliar environments such as urban areas can prove to be a formidable task. Thus, accurately estimating target location and achieving robust target tracking in complex urban environments become even more challenging due to the scarcity of essential signal propagation parameters in unfamiliar territories.

The limitation of power capacity and power inefficiency also poses a substantial challenge in the realm of UAVs [7]. UAVs inherently operate with finite battery power, and prolonged tracking operations can rapidly deplete their power reserves. Inefficient power consumption strategies further exacerbate this problem, resulting in reduced mission durations, frequent requirements for recharging or battery replacement, and increased operational costs. Addressing these challenges becomes crucial for improving the overall effectiveness and sustainability of UAV-based tracking systems.

In real-time target tracking, delay poses another significant challenge [8]. The timely decision-making and response are compromised when delays occur in detecting and updating target positions. These delays can result from factors such as communication latency, processing time, and inefficient coordination between the entities of the tracking system, all of which can hinder the system's ability to respond effectively. Communication delays occur due to the time required for the information to travel between the UAVs and other network components, including the ground sensor nodes and base stations. Processing time refers to computational delays incurred during data processing, target position estimation, and decision making. Inefficiencies in coordination can result from suboptimal task allocation, communication protocols, or synchronization mechanisms among entities. These delays collectively impact the real-time nature of the tracking system, potentially compromising its effectiveness in dynamic scenarios where prompt responses are crucial.

Furthermore, ensuring precise estimation of target position and, consequently, achieving accurate tracking represent critical challenges in the field of target tracking [9]. Precise prediction of the future movements of targets is essential for maintaining effective tracking, especially when dealing with high-speed targets or intricate maneuvers. Moreover, the presence of obstacles like buildings and trees can obstruct the UAV's line of sight, resulting in incomplete or imprecise target observations. Several factors, including sensor noise, uncertainties in target behavior, obstructions, and complex surroundings, significantly impact the precision of the estimate of the target's position. Consequently, these factors further complicate the task of tracking specific targets.

Motivated by the challenges encountered in tracking targets with multiple UAVs in urban environments, this paper introduces a robust approach to address these issues. Our strategy involves designing a comprehensive system model that integrates mobile sensor nodes (SNs), Evolved/Next Generation NodeBs (e/gNBs) as part of the Radio Access Network (RAN) on the ground, and UAVs, leveraging the principles of the Open Radio Access Network (Open-RAN). The Open-RAN is chosen for its flexibility and open standards, which facilitate efficient communication and coordination between UAVs and network components. This model aims to improve target tracking accuracy, optimize power efficiency, and improve real-time performance. We focus on maximizing tracking accuracy, reducing update delays, and minimizing UAV power consumption. Furthermore, considering the presence of urban obstacles, such as buildings, our approach highlights the need for effective collision avoidance strategies. By leveraging Open-RAN's capabilities, we seek to address these challenges and improve the overall tracking performance while conserving UAV resources.

To explore these trade-offs and meet the objectives, our approach integrates a range of advanced algorithms and techniques, including particle filtering (PF), Q-learning, clustering, multilateration, normalization, and a path planning algorithm. The PF algorithm enhances target position estimation accuracy, while multi-lateration provides precise target localization using RSS information. The Q-learning algorithm allows UAVs to intelligently select optimal flight directions taking into account accuracy, delay, and power consumption. To further optimize UAV performance, we incorporate a path planning algorithm to determine the best route for UAVs to approach and track the target effectively. Clustering techniques facilitate coordination among UAVs, improving overall tracking efficiency. As described in [10], the clustering technique (central network) performs better in terms of power consumption. In our multi-criteria decision-making process, we employ Min-Max normalization to standardize diverse ranges of accuracy, delay, and power values, ensuring fair and balanced evaluation. This normalization method enables an effective comparison of various criteria, facilitating informed decision-making. By integrating advanced algorithms and techniques and using Open-RAN for improved communication, the challenges of tracking the target in urban environments are addressed, improving both the tracking performance and the efficiency of resources.

Through the integration of these advanced algorithms, our approach seeks to revolutionize target tracking systems and improve their accuracy, power efficiency, and real-time performance in ground-based scenarios. The main contributions of this study are given below.

1- Developing a PF-based localization algorithm utilizing RSS to estimate the position of a moving target, improving tracking accuracy and reliability through collaboration among mobile sensor nodes.
2- Developing a multi-agent Q-learning algorithm for UAV swarm tracking, focusing on real-time performance, accuracy, power efficiency, and avoidance of

obstacles collisions, optimizing decision-making and navigation.

3- Developing an offloading strategy is introduced to address UAV power limitations considering priority, computation time, transmission delay, and queueing delay, to ensure efficient resource allocation and optimal tracking performance.

The remainder of this paper is organized as follows. Section II provides an overview of the related work on cooperative sensing and target tracking using UAVs. Section III presents the system model and formulation of the power, delay, and accuracy trade-offs. Section IV provides the formulation of the problem. Section V details the proposed cooperative sensing framework, including collaboration strategies and resource allocation techniques. Section VI presents the experimental setup and discusses the obtained results. Finally, Section VII concludes the paper and highlights future research directions in the field of cooperative sensing with multi-UAVs for target tracking.

II. Related Work

In recent years, the deployment of multiple UAVs has emerged as a promising approach to tackle the challenges associated with target tracking and search operations. Various studies have explored different methodologies to improve the performance of multi-UAV systems in these domains. For example, in [11] an online reinforcement learning algorithm is introduced that simultaneously addresses target tracking while considering energy refueling needs, showcasing a novel way to optimize UAV operations under energy constraints.

In [12], a structured learning-based graph matching method is developed specifically for tracking applications. This method is divided into two layers, effectively addressing both non-stationary issues and routing problems, thereby improving the reliability of tracking in complex environments. In [13] a mixture model is proposed that refines prior probability distributions, aiding UAVs to maintain effective tracking despite environmental fluctuations. In [14] a dynamic labor distribution-based ant colony algorithm is used that demonstrated a high level of self-organization. This approach allows for rapid responses and flexibility when navigating dynamic environments, showcasing the potential of bioinspired algorithms in enhancing multi-UAV tracking systems.

In [15], an autonomous UAV swarm system is introduced to track mobile RF targets using omnidirectional RSS sensors. To handle dynamic channel conditions such as varying transmission power, the flight decision process was modeled as a constrained Markov decision process (CMDP). An enhanced multi-agent reinforcement learning (MARL) algorithm was proposed to coordinate the UAVs, reducing redundant paths and improving tracking accuracy. Simulations showed that the approach outperforms standard Q-learning in terms of search time and localization success.

In [16], an efficient Vision Transformer (ViT)-based framework, TATrack, has been proposed for real-time UAV tracking. This framework combines feature learning and template-search coupling within a one-stream ViT model, avoiding the need for an additional relation modeling module. They introduced a method to maximize mutual information (MI) between the template image and its feature representation. Furthermore, a novel MI-based knowledge distillation technique was introduced to balance accuracy and efficiency. Extensive evaluations of five benchmarks demonstrate that TATrack achieves state-of-the-art performance in UAV tracking tasks.

III. Preliminaries

Here, we define the entities involved in the proposed scheme, as well as the threat models and security requirements and goals that we aim to achieve.

A. System Model

The proposed cooperative sensing framework for target tracking involves creating a comprehensive system model that ensures accurate and efficient tracking. This model integrates various entities, each playing a vital role in the tracking process. These entities include Target Nodes (TNs), Sensor Nodes (SNs), e/gNBs, and UAVs. In addition, we leverage the Open-RAN architecture to improve network coordination and communication efficiency.

In this framework, TNs are the moving entities on the ground that need to be tracked, such as vehicles, pedestrians, or other mobile objects. SNs are strategically deployed to detect TNs and estimate their positions using techniques such as RSS and PF. These SNs provide the initial position estimates needed for accurate tracking. Unlike static deployments, the SNs in our model are mobile, allowing them to adjust their positions dynamically to improve detection accuracy and coverage. Their movement is optimized to maintain an effective sensing range while minimizing localization errors, ensuring more precise and reliable target tracking. To facilitate communication between SNs and UAVs, e/gNBs are deployed throughout the environment, and each e/gNB manages a specific area. In the Open-RAN architecture, e/gNBs are intelligently managed by the RAN Intelligent Controller (RIC), which optimizes network performance and coordination between various elements [17]. The RIC operates in both near-real-time (near-RT) and non-real-time (non-RT) modes, handling tasks like resource management and interference mitigation, while also enabling long-term optimizations. In this system, the near-RT RIC coordinates UAV clusters through specialized applications (xApps), allowing real-time adjustments to UAV flight paths and communication strategies. The e/gNBs receive target position data from SNs and use selection algorithms to assign the most suitable UAVs for tracking based on their proximity and availability. Additionally, e/gNBs have detailed knowl-

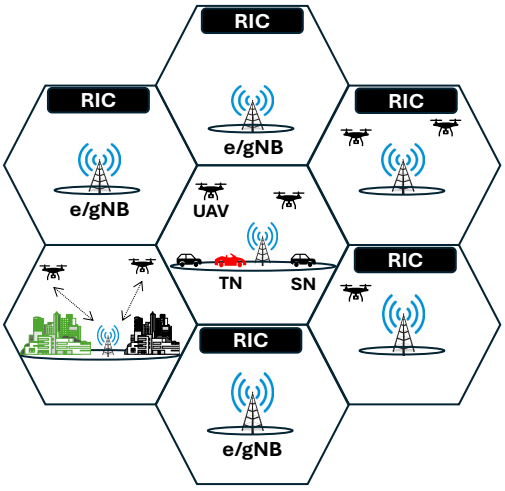


Figure 1: The system model designed within an Open-RAN environment, where a RIC supports single/multiple e/gNBs.

edge of obstacles in their areas. Given the limited memory capacity of UAVs, they obtain obstacle information from e/gNBs as they move. When a UAV enters a new area, it receives updated obstacle information from the corresponding e/gNB, ensuring efficient memory use and enabling UAVs to track targets effectively and safely.

UAVs, equipped with omnidirectional RSS sensors, are crucial to the tracking process. The e/gNBs assign UAVs to each target for localization and tracking, using the RSS technique alongside the Q-Learning algorithm. To ensure accurate target position estimation, at least three UAVs are dedicated to each target. This setup allows for improved localization precision and enhances overall tracking effectiveness. The assigned UAVs work collectively as a cluster with one head of the cluster (CH), collaborating to track the target. By forming a cluster, UAVs can leverage their combined capabilities and effectively address challenges such as localization accuracy, obstacle avoidance, and real-time tracking.

Figure 1 illustrates a schematic representation of the system model in an urban environment, which includes three UAVs, two e/gNBs, a TN represented by a red vehicle, and additional vehicles that serve as SN.

In this system, it is assumed that there are multiple targets, denoted $m = 1, \cdots, M$, with ground mobility and a total of N UAVs, denoted agents that fly at different altitudes. The location of the m-th RF target at any given time t is indicated by $TN_{m,t} = (x_{m,t}, y_{m,t})$. The velocity of movement of the target is defined as $TN^{v}_{m,t} = (v_{x,m,t}, v_{y,m,t})$, representing the velocity of the target along the x and y axes at time t. The time-varying location of the n-th UAV at time t is also denoted as $U_{n,t} = (\dot{x}_{n,t}, \dot{y}_{n,t}, \dot{z}_{n,t})$, and its flight velocity is defined as $U^{v}_{n,t} = (\dot{v}_{x,n,t}, \dot{v}_{y,n,t}, \dot{v}_{z,n,t})$. In our system, each UAV is equipped with a range sensor that enables it to measure the distance between itself and the TN m. These distance measurements are continuously obtained at every timestamp as the UAVs fly in the vicinity of the ground targets. By incorporating range sensors into the UAVs' capabilities, we can gather precise and real-time information about the distances between the UAVs and the targets they are tracking. This distance measurement data plays a crucial role in estimating the positions and movements of the targets, allowing for effective target tracking and localization by the UAV swarm. The classic narrowband radio propagation path loss model provides a non-linear equation that relates the RSS to the distance for distance measurement based on location. The equation can be expressed as follows [18]:

$$PL_{mn,d,t} = PL_{mn,0} + 10\,\alpha\, log_{10}\, d_{mn,t} + \eta_{mn,t} \quad (1)$$

where $d_{mn,t}$ is the distance between TN m and UAV n. $\eta_{mn,t}$ is a Gaussian random variable representing log-normal shadow fading effects in multipath environments, $PL_{mn,0}$ denotes the signal power loss in units of decibels (dB) at a reference distance of 1 meter [18], while $PL_{mn,d}$ represents the signal power loss at a distance $d \geq 1$ meter. α is the path loss exponent that determines the rate at which signal power decreases with distance. Here, $PL_{mn,d,t} = P_{TX,t} - P_{RX,t}$ can be measured by the received and transmitted signal power $P_{RX,t}$, $P_{TX,t}$ at time t. P_{TX} represents the target's constant transmit power, which is known to the UAVs [15], [19].

Given the presence of obstacles in the urban environment, it is crucial to consider both LoS and NLoS conditions when modeling path loss. To account for the measurement noise, we model it as follows [20]:

$$\begin{aligned}\eta_{mn,t} \sim \nu_{mn,t}\, \mathcal{N}\left(0, \sigma^2_{LoS}\right) \\ + (1 - \nu_{mn,t})\, \mathcal{N}\left(\mu_{NLoS}, \sigma^2_{NLoS}\right)\end{aligned} \quad (2)$$

where ν_{mn} reflects the probabilities of a direct, unobstructed path between the UAV n and the TN m. The characteristics of the LoS measurements are modeled by a zero mean Gaussian distribution, denoted as $\mathcal{N}\left(0, \sigma^2_{LoS}\right)$, where the variance is σ^2_{LoS}. This implies that the LoS measurements exhibit a symmetric distribution around zero, with a spread determined by the variance. In contrast, the statistical profile of the NLoS measurements is described by another Gaussian distribution, denoted $\mathcal{N}\left(\mu_{NLoS}, \sigma^2_{NLoS}\right)$. This distribution has a mean μ_{NLoS} and a variance σ^2_{NLoS}. NLoS measurements capture the effects of signal reflections, diffraction, and scattering caused by obstacles in the environment. The mean and variance of this distribution provide valuable information on the average value and variability of the NLoS measurements, respectively. By considering these statistical profiles, we can better understand and characterize the nature of the measurements obtained in both LoS and NLoS scenarios. This information is crucial for developing accurate models and algorithms for target tracking and localization using RSS measurements in complex urban environments.

The parameter ν_{mn} plays a crucial role in determining the probability of signal received by the UAV n from TN m under the LoS condition. As mentioned in [15], ν_{mn}

is determined by the elevation angle $\theta_{mn,t}$ between UAV n from TN m, a_t, and b_t as follows:

$$\nu_{mn,t} = \frac{1}{1 + a_t e^{-b_t(\theta_{mn,t} - a_t)}} \tag{3}$$

The parameters a_t and b_t are used to quantify specific characteristics of the urban environment at time t. a_t represents the ratio of the structured area, which includes buildings and other man-made structures, to the total land area. It provides information on the extent of urbanization and the density of structures in the environment. A higher value of a_t indicates a higher concentration of built-up areas compared to open spaces. On the other hand, b_t denotes the number of buildings per unit area of land. It provides information about the building density within the urban environment. A higher value of b_t indicates a greater concentration of buildings in a specific area, signifying a more densely populated or urbanized region. By incorporating these parameters into our analysis, we can better understand the characteristics of the urban environment, such as the level of urbanization and the density of buildings. This information is essential for studying various aspects, such as signal propagation, obstacle avoidance, and overall system performance in urban target tracking scenarios.

B. Integration of Particle Filtering Algorithm and RSS

The PF algorithm is a method used to estimate the state of a system based on sequential observations. It consists of several steps to iteratively update and refine the estimate of the state of the system. The steps of the PF algorithm are as follows [21]:

- Initialization: Initially, a set of particles is generated from a prior distribution, representing possible target positions at the initial time step. Each particle is assigned an equal weight.
- Prediction: At each time step, the particles are propagated according to a motion model that describes the expected movement of the target. This prediction step updates the positions of the particles based on the motion model.
- Update: The weights of the particles are updated on the basis of the observed measurement at the current time step.
- Resampling: After the weights are updated, resampling is performed to select particles for the next iteration. Particles with higher weights have a higher chance of being selected, while particles with lower weights are less likely to be chosen. This resampling step helps maintain a diverse set of particles and focuses the estimation on regions with higher likelihoods.

The integration of the PF algorithm with RSS measurements provides a significant improvement in the accuracy and reliability of the estimation of the position of the target [22]. This integration involves incorporating additional information from RSS measurements into the PF algorithm. In this integration, various notation is considered. The state of the system is represented by the true position of the target at a given time, denoted as x_t, where t represents the time step. The observation at each time step is the RSS measurement, denoted as z_t. The PF algorithm uses particles, where each particle i represents a hypothesis or possible location of the target, denoted x_t^i. Finally, the weight associated with each particle i is calculated to represent the probability that the particle is the true target position, denoted as w_t^i.

In this integration, the RSS measurement is incorporated into the update step of the algorithm. The likelihood of each particle's position given the RSS measurement is typically calculated using a Gaussian likelihood function, which considers the difference between the observed RSS and the expected RSS at each particle's position. In PF algorithm, the weight update equation can be expressed as:

$$w_t^i \propto w_{t-1}^i \times \mathcal{P}(z_t | x_t^i) \tag{4}$$

where w_{t-1}^i is the weight of the particle at the previous time step, and $\mathcal{P}(z_t|x_t^i)$ represents the likelihood of the observed RSS measurement given the particle's position.

By iteratively performing the prediction, update, and resampling steps, the PF algorithm effectively combines information from the motion model and the observed RSS measurements to accurately estimate the target's position.

C. Multi-Agent Q-learning Algorithm

The multi-agent Q-learning algorithm [15] is an extension of the classical Q-learning algorithm [23] that enables multiple agents to learn and make intelligent decisions in a dynamic environment. It is a reinforcement learning technique in which each agent interacts with the environment, observes its state, selects actions, and learns from the rewards received. The algorithm aims to find the optimal policy for each agent to maximize its cumulative rewards over time.

At each time step, each agent selects an action based on its current state and the values stored in its Q-table. The Q-table contains Q-values, which represent the expected cumulative rewards for taking a specific action in a given state. The Q-values are updated iteratively using the Bellman equation [24]:

$$\begin{aligned} Q(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \\ \alpha \left[R(s,a) + \gamma \max_{a' \in \mathring{A}} Q(s',a') \right] \end{aligned} \tag{5}$$

where $Q(s,a)$ is the Q-value for state s and action a, $\mathring{A}$ is the set of all admissible actions, α is the learning rate that determines the weight given new information, $R(s,a)$ is the reward received after taking action a in state s, $\gamma \in (0,1]$ is the discount factor that balances the importance of future rewards, s' is the next state, and $a' \in \mathring{A}$ is the next action.

The Q-value update equation combines the current Q-value with the discounted maximum Q-value of the next state-action pair, scaled by the learning rate. This update

rule allows agents to learn from the rewards they receive and update their action selection policies accordingly. The multiagent Q-learning algorithm involves coordination and cooperation among the agents. Each agent updates its Q-values based on its own experiences and the observations of other agents' actions and rewards. The algorithm can be applied to various multi-agent systems, such as multi-robot systems, where agents need to collaborate and communicate to achieve common goals.

D. Offloading Strategy

In order to efficiently manage the task offloading process in the multi-UAV system, we propose an offloading strategy that considers the remaining power of the UAVs. This strategy aims to optimize the utilization of available resources while ensuring effective task allocation and minimizing power consumption. The strategy involves the following steps:

1. Threshold setting: The offloading strategy uses two threshold values, $PW^{(w)}$ and $PW^{(e)}$. These thresholds are predefined on the basis of the remaining power of the UAV. $PW^{(w)}$ represents the threshold at which the power of the UAV reaches a low level, indicating the need for partial offloading. $PW^{(e)}$ represents the threshold at which the power of the UAV reaches a critical level, necessitating complete offloading and initiating the landing process.

2. Task offload decision: At each timestamp, the UAV assesses its remaining power $PW_{U,t}$ and compares it with the threshold values $PW^{(w)}$ and $PW^{(e)}$. Based on this comparison, a decision is made regarding the task-offloading strategy.

3. Partial offloading: When the remaining power of the UAV is equal to or below $PW^{(w)}$, it indicates a low power level. In this case, the UAV selectively offloads some of its tasks to its cluster head (CH) and/or nearby e/gNBs. This partial offloading strategy allows for the distribution of computational load and ensures the efficient utilization of available resources. Additionally, the UAV sends a warning message to both the CH and nearby e/gNBs, notifying them about its power status.

4. Complete offload and landing: If the remaining power of the UAV reaches or falls below $PW^{(e)}$, it indicates a critical power level that requires immediate action. In this situation, the UAV initiates complete task offloading, transferring all its tasks to the CH and/or nearby e/gNBS. During the flight, the UAV sends an error message to both the CH and nearby e/gNBs, indicating its power status and triggering the landing process.

The decision-making process can be formulated as follows:

$$\begin{cases} \text{Partial Offloading} & \text{if } PW_{U,t} \leq PW^{(w)} \\ \text{Complete Offloading and Landing} & \text{if } PW_{U,t} \leq PW^{(e)} \end{cases} \tag{6}$$

By dynamically adjusting the task offloading strategy based on the remaining power of UAVs, this approach ensures efficient resource utilization, minimizes power consumption, and maintains the operational efficiency of the multi-UAV system.

IV. Problem Formulation

In this work, we focus on continuously estimating and tracking a target's position using a swarm of UAVs. Our approach prioritizes accurate target tracking while minimizing both power consumption and delay, ultimately enhancing the efficiency and effectiveness of the multi-UAV tracking system.

A. Tracking Accuracy

Assuming that CR_n represents the coverage radius of the n-th UAV, it is necessary for a TN to be within the coverage zone of at least three UAVs. In other words, the horizontal distance between the TN and each UAV, denoted as r_n, must be less than or equal to the respective coverage radius R_n, where $n = 1, 2, \cdots, N$ and $N \geq 3$ is the number of available UAVs. Therefore, the first constraint to successfully localize the node can be expressed as $r_n \leq CR_n$. Given the estimated horizontal distance $\hat{r}_{mn}$ and the known projection $(\dot{x}_{n,t}, \dot{y}_{n,t})$ of the n-th UAV at time t, the position of m-th TN can be estimated by finding the coordinates $(\hat{x}_{m,t}, \hat{y}_{m,t})$ that establishes

$$\hat{TN}_{m,t} = (\hat{x}_{m,t}, \hat{y}_{m,t}) = \underset{x,y}{\operatorname{argmin}} \left\{ \sum_{n=1}^{N} \left(\|U_{n,t} - TN_{m,t}\| - \hat{r}_{mn} \right)^2 \right\} \tag{7}$$

where $\|.\|$ is the Euclidean distance. It provides a straightforward measurement of the spatial separation between the two entities for example, the n-th UAV and the m-th target node. Here, $\hat{r}_{mn}$ is the estimated horizontal distance between the UAV n, and TN m.

$$\|U_{n,t} - TN_{m,t}\| = \sqrt{\left(\dot{x}_{n,t} - x_{m,t}\right)^2 + \left(\dot{y}_{n,t} - y_{m,t}\right)^2} \tag{8a}$$

$$\hat{r}_{mn} = \sqrt{\hat{d}^2_{mn,t} - \dot{z}^2_{n,t}} \tag{8b}$$

$$\mathsf{A} = \hat{d}_{mn} = \left\| U_{n,t} - \hat{TN}_{m,t} \right\| \tag{8c}$$

This formulation provides the target nodes' position for a given $\dot{z}_{n,t}$ at time t. Given the estimated position, the localization error is given by:

$$\mathsf{E} = \sqrt{\sum_{n=1}^{N} \left| \hat{r}_{mn,t} - r_{mn,t} \right|^2} \tag{9}$$

Given the impact of UAV altitude $\dot{z}$ on minimizing localization error, it is necessary to determine the optimal altitude $\dot{z}$. To this end, we formulate the problem as an optimization task. The objective is to determine the

altitude value that results in the lowest localization error. This can be expressed mathematically as follows:

$$\textbf{P1:} \quad \min_{\dot{z}} \quad \{\mathsf{E}\} \quad \text{subject to} \quad (8b) \tag{10}$$

The optimization problem described in Equation (10) is influenced by various relevant parameters that are vital for the localization process. These parameters encompass the number of UAVs (N), the inter-UAV distance, and the range and coverage capabilities of the UAVs denoted as CR_n. As the localization is achieved using the RSS technique, the optimization problem takes into account the effects of path loss, attenuation, reflection, diffraction, and scattering caused by obstacles obstructing the line of sight between the UAVs and target nodes. Furthermore, the sensor characteristics of the UAVs, the properties of the target being localized, and the communication constraints among the UAVs also contribute to these optimization considerations. Each of these parameters plays a significant role in determining the localization performance and must be carefully considered in formulating the optimization problem. By accounting for these parameters, the objective of the optimization problem is to identify the optimal altitude $\dot{z}$ that minimizes the localization error. This ensures the achievement of accurate and reliable target localization and tracking by the multi-UAV system.

B. Power Consumption

In this system, the power consumption of UAVs poses a significant challenge due to their limited power capacity. Efficient management of power resources becomes crucial for successful target tracking operations. To address this challenge, we propose a task-offloading strategy in this work. The strategy involves making decisions on whether to execute a task locally on a UAV or offload it, either partially or fully, to other entities in the network such as other UAVs or e/gNBs. By intelligently distributing the computational load, we aim to optimize power consumption and enhance the overall performance of the target tracking system. This strategy enables the system to effectively utilize available resources and mitigate the impact of power limitations on UAVs, ultimately improving the system's power efficiency and tracking capabilities.

As explained in [25], the power consumption for an UAV during target tracking is the sum of computational power PC_{loc}, task offloading or communication PC_{off}, and mobility or flight PC_{fly} as follows:

$$\mathsf{P}_{uav} = PC_{loc} + PC_{off} + PC_{fly} \tag{11}$$

The computation of a task performed locally on a UAV depends on several factors. These factors include task complexity O, the number of bits of input data in the computation I_I, the size of the offload data I_O, and the power consumption for each CPU cycle in the UAV $PC_{U_{cpu}}$. The power consumption for UAV running each CPU cycle can be expressed as follows:

$$PC_{loc} = \begin{cases} O_T \times PC_{U_{cpu}} \times I_I & \text{Fully locally} \\ O_T \times PC_{U_{cpu}} \times (I_I - I_O) & \text{Partially locally} \end{cases} \tag{12}$$

Besides, the power consumption for offloading the task by an UAV to e/gNB/another UAV can be written as [26]:

$$PC_{off} = P_{TX} \times \frac{I_O}{R_{U2X}(P_{TX})} \tag{13}$$

where P_{TX} denotes transmit power and $R_{U2X}(P_{TX})$ in bits-per-second (b/s) denotes the offloading rate from an UAV to another entity.

When analyzing the impact of UAV mobility on power consumption, several parameters need to be considered, including speed, payload, horizontal movement, vertical movement, and hovering [27]. These factors play a significant role in determining the power expenditure associated with the mobility of UAVs. The cumulative power consumption for mobility/flight can be mathematically represented as follows:

$$PC_{fly} = PC^{(1)}_{\dot{v},\Delta t} + PC^{(2)}_{\mathsf{m},\Delta t} + PC^{(3)}_{\dot{v},\mathsf{m},\Delta t} + PC^{(4)}_{\dot{v},\mathsf{m},\Delta t} + PC^{(5)}_{\dot{v},\mathsf{m},\Delta t} \tag{14}$$

where

$$PC^{(1)}_{\dot{v},\Delta t} = \mathsf{k}\, v^3 \tag{15a}$$

$$PC^{(2)}_{\mathsf{m},\Delta t} = \mathsf{k}\, \mathsf{m} \tag{15b}$$

$$PC^{(3)}_{\dot{v},\mathsf{m},\Delta t} = \frac{1}{2}\rho\, \mathsf{a}\, C_d\, \dot{v}_x^3 + \mu\, \mathsf{m}\, \mathsf{g}\, \dot{v}_x \tag{15c}$$

$$PC^{(4)}_{\dot{v},\mathsf{m},\Delta t} = \mathsf{m}\, \mathsf{g}\, \dot{v}_y \tag{15d}$$

$$PC^{(5)}_{\dot{v},\mathsf{m},\Delta t} = \mathsf{m}\, \mathsf{g}\, \dot{v}_h \tag{15e}$$

here, $PC^{(1)}_{\dot{v},\Delta t}$ represents the power consumed when the UAV flies at a certain speed in meters per second. $PC^{(2)}_{\mathsf{m},\Delta t}$ refers to the power consumed when the UAV flies with a payload m in grams. $PC^{(3)}_{\dot{v},\mathsf{m},\Delta t}$ represents the power consumed when the UAV flies in a straight horizontal line. $PC^{(4)}_{\dot{v},\mathsf{m},\Delta t}$ is the power consumed when the UAV flies vertically to reach a desired altitude. Lastly, $PC^{(5)}_{\dot{v},\mathsf{m},\Delta t}$ denotes the power consumed when the UAV hovers at a specific altitude. m is the mass of the UAV, g is the acceleration due to gravity, $\dot{v}$ is the velocity of the UAV. ρ is the air density, a is the frontal area of the UAV, C_d is the drag coefficient, μ is the coefficient of friction, k represents a coefficient that respectively relate velocity and payload to the power consumption.

Given our objective of minimizing UAV power consumption, we can formulate the problem as follows. We aim to minimize the overall power consumption of the UAV, which consists of communication-related power consumption, computation-related power consumption, and mobility-related power consumption. This objective is subject to constraints such as task computation, task allocation, transmit power allocation, and UAV trajectory design. By optimizing these variables and their allocations, we can achieve the goal of reducing power

consumption while ensuring efficient task execution and communication among UAVs and other entities in the network. This problem can be formulated as follows:

$$\textbf{P2:} \min_{\mathbf{L},\mathbf{O},\mathbf{F}} \quad \{\mathsf{P}_{uav}\}$$
$$\text{subject to (12), (13), (15a), (15b), (15c), (15d), (15e)} \tag{16}$$

where $\mathbf{L} = \{O_T, EC_{U_{cpu}}, I_I, I_O\}$, $\mathbf{O} = \{P_{TX}, R_{U2X}\}$, and $\mathbf{F} = \{\mathsf{m}, \dot{v}, \mathsf{a}\}$. In Equation (16), we aim to minimize the power consumption of a UAV. The optimization problem explicitly depends on several factors, including the velocity of the UAV ($\dot{v}$), the mass of the UAV (m), the transmit power (P_{TX}), the offload rate (R_{U2X}), the distance to the target, the distance to the CH and the distances to nearby e/gNBs.

The objective is to find the optimal values of these parameters that minimize the power consumption, taking into account the constraints and considerations specific to the UAV's operation. By optimizing these parameters, we can effectively reduce the power consumption of the UAV, thus increasing its operational efficiency and extending its flight time.

C. Delay

In real-time target tracking systems, delay also poses a significant challenge. This delay can include various components such as transmission delay, computing delay, and propagation delay. These delays can affect the overall performance of the system, affecting the accuracy and efficiency of the target tracking process. Minimizing delay is crucial to ensure timely and accurate updates in the tracking information, enabling UAVs to make informed decisions and respond swiftly to changes in the target's position or movement. Therefore, in the design and implementation of such systems, reducing and managing delay becomes a critical consideration. Therefore, the total delay can be formulated as follows:

$$\mathsf{D}_{total} = D_{tran} + D_{prop} + D_{comp} \tag{17}$$

where D_{tran}, D_{prop}, and D_{comp} refer to transmission delay, propagation delay, and computing delay, respectively.

The transmission delay in the target tracking system refers to the time it takes for data to travel from one point to another within the communication network. In the context of UAV target tracking, transmission delay can arise when the UAVs need to exchange information, such as target positions or tracking updates, with each other or with the central control station. The transmission delay can be quantified using the following equation:

$$D_{tran} = \frac{Data_{size}}{BW} \tag{18}$$

where $Data_{size}$ is data size to be transmitted (in bits) and BW is transmission bandwidth (in bits per second). In a multi-UAV tracking scenario, UAVs might need to send and receive data to coordinate their tracking efforts. For example, UAVs can share their estimated target positions or other relevant information to ensure comprehensive tracking coverage. The transmission delay becomes crucial in such cases to avoid delays in data exchange, which could lead to inaccurate or outdated target tracking information among UAVs.

The propagation delay in the target tracking system refers to the time it takes for signals or data to propagate through the communication medium from the transmitter to the receiver. In the context of UAV target tracking, propagation delay can arise due to the finite speed at which electromagnetic waves travel through the air or other communication channels. The propagation delay can be quantified using the following equation:

$$D_{prop} = \frac{Dist}{S_{prop}} \tag{19}$$

where $Dist$ is distance between the transmitter and receiver (in meters) and S_{prop} refers to propagation speed of the signal (in meters per second). The propagation delay depends on the distance between the transmitter (e.g., one UAV) and the receiver (e.g., another UAV or a ground station) and the speed at which the signal travels through the communication medium. Since the speed of light is constant in the air (approximately 3×10^8 meters per second), the propagation delay depends mainly on the distance between the communicating entities. In the multi-UAV tracking scenario, the UAVs might be spread out across a certain area, and the propagation delay becomes a consideration when exchanging data or control signals between UAVs or between UAVs and e/gNB. Longer distances between UAVs or the UAVs and e/gNB can lead to longer propagation delays, which may affect the real-time tracking and coordination of the UAVs.

Computing delay on a UAV in the target tracking system refers to the time it takes for the UAV's onboard processing unit (e.g., CPU) to perform computations and make decisions. This delay can occur when the UAV needs to process sensor data, perform calculations, and execute control algorithms to determine its next actions. The computing delay can be expressed as follows:

$$D_{comp} = \frac{CPU_y}{CPU_l} \tag{20}$$

where CPU_y refers total number of CPU cycles required to perform the computation and CPU_l denotes CPU clock speed (in cycles per second). The computing delay depends on the complexity of the computations required for target tracking and the speed of the UAV's CPU. More complex computations or higher CPU clock speeds can result in shorter computing delays, allowing the UAV to process information and make decisions more quickly. In a multi-UAV target tracking system, computing delays are critical to consider, especially when multiple UAVs need to collaborate and exchange information. Delays in processing data or making decisions could lead to reduced tracking accuracy, potential collisions, or inefficiencies in the UAV's movements.

Given our objective, minimizing delay, we formulate the problem as follows:

$$\textbf{P3:} \quad \min_{Data_{size}, Dist, CPU_l} \{\mathsf{D}_{total}\} \quad \text{subject to (18), (19), (20)} \tag{21}$$

This objective is subject to transmission, propagation and computing delay. The optimization problem explicitly depends on the size of the data $Data_{size}$, the distance between sender and receiver $Dist$, and the CPU power CPU_l. The objective is to find the optimal value for the size of the exchanged data and the optimal distance between UAVs/between UAVs and e/gNBs.

V. Our Approach: Design of an Approach for Target Tracking by Multi-UAVs

This section presents our approach to target tracking, integrating key algorithms for accuracy and efficiency. A PF-based localization algorithm is used by a swarm of SNs to estimate the target's position. e/gNBs then select suitable UAVs for tracking based on these data. A clustering technique forms UAV clusters, with at least three UAVs and one designated as the Cluster Head (CH). To optimize tracking, UAVs use a Q-learning algorithm to determine the best flight path, considering power, accuracy, delay, and collision avoidance. Detailed descriptions of these methods are provided in the following sections.

A. PF-based Localization Algorithm

Localization based on RSS measurements poses various challenges, such as non-linear of sight (NLoS) and multipath propagation, which can lead to inaccurate distance estimations. The susceptibility of RSS measurements to variability and noise further adds uncertainty to the localization process. In addition, the acquisition of a sufficient number of RSS measurements from reference points or anchors can be difficult in certain scenarios.

To address these challenges, PF emerges as a powerful technique in RSS-based localization. PF addresses these issues by effectively integrating information from the motion model, RSS measurements, and particle weights. This integration creates a versatile and robust framework that enables accurate estimation of the target's position, even in the presence of NLoS effects, noise, sparse measurements, and dynamic target movement. By incorporating RSS into the PF algorithm, it significantly improves the precision and reliability of the target position estimation.

Consider a scenario in which a mobile target node is located in an unknown location $TN = (x, y)$, while n mobile SNs are located in known locations $SN_j = (\ddot{x}_j, \ddot{y}_j)$, with $1 \leq j \leq n$ in a 2D area. A SN swarm will use a PF algorithm to infer TN_t given the signal power received $P_{RX,t}$ and the signal power transmitted $P_{TX,t}$ at time t from the target node. This estimation process involves determining the belief or probability distribution, denoted as $bel\,(TN_t)$, which represents the confidence in the position of the target node at time t. We can express this belief as $\mathcal{P}\,(TN_t|P_{RX,1:t}, P_{TX,t})$, where $P_{RX,1:t}$ represents the signal power received over the time interval $[1:t]$. By employing the Bayesian approach, we can derive the posterior as follows [28]:

$$\begin{aligned} &bel\,(TN_t) \\ &= \mathcal{P}\,(TN_t|P_{RX,1:t}, P_{TX,t}) \\ &= \frac{\mathcal{P}\,(P_{RX,t}|TN_t, P_{TX,t})\,\mathcal{P}\,(TN_t|P_{RX,1:t-1}, P_{TX,t-1})}{\mathcal{P}\,(P_{RX,t}|P_{RX,1:t-1})} \\ &= \vartheta \mathcal{P}\,(P_{RX,t}|TN_t, P_{TX,t})\,\mathcal{P}\,(TN_t|P_{RX,1:t-1}, P_{TX,t-1}) \end{aligned} \tag{22}$$

where, according to Equation (1), $P_{RX,t} = P_{TX,t} - L_0 - 10\,\alpha\,log_{10}d - \eta_t$ and ϑ represents the normalized constant that ensures the sum of $\mathcal{X} = \mathcal{P}\,(TN_t|P_{RX,1:t-1}, P_{TX,t-1})$ over all possible values of TN_t equals to 1. Applying the Chapman-Kolmogorov equation [29], we obtain the following result:

$$\mathcal{X} = \int \mathcal{P}\,(TN_t|TN_{t-1})\,\mathcal{P}\,(TN_{t-1}|P_{RX,1:t-1}, P_{TX,t-1})\,dTN_{i,t-1} \tag{23}$$

where the Chapman-Kolmogorov equation states that the probability of transitioning from one state to another at a future time step can be calculated by considering the intermediate states. Mathematically, it can be expressed as:

$$\begin{aligned} &\mathcal{P}(X_{t+1}|X_0, \ldots, X_t) \\ &\quad = \int \mathcal{P}(X_{t+1}|X_t) P(X_t|X_0, \ldots, X_{t-1}) dX_t \end{aligned}$$

The posterior probabilities are iteratively obtained using the following recursive formulation:

$$bel\,(TN_t) = \mathcal{Z} \int \mathcal{P}\,(TN_t|TN_{t-1})\,bel\,(TN_{t-1})\,dTN_{t-1} \tag{24}$$

where

$$\mathcal{Z} = \vartheta \mathcal{P}\,(P_{RX,t}|TN_t, P_{TX,t})$$

Based on Equation (24), we can outline the localization principle through the following steps. The initial belief of the location of the target, denoted as a set of particles S, serves as the starting point for the localization process. The evolution of these particles plays a crucial role in achieving an accurate and convergent representation of the target's positions over time. To capture the transition of the target's movement, samples are drawn from the motion transition model $\mathcal{P}(TN_t|TN_{t-1})$ to represent the distribution of target positions in the next time instance. The transition model $\mathcal{P}(TN_t|TN_{t-1})$ describes the probability that a target node transitions from position TN_{t-1} to position TN_t, indicating how the target's location evolves over time. This allows us to track the target's trajectory and estimate its location at each time step.

During the particle filtering weighting phase, we assess the compatibility of the particles generated from the transition model $\mathcal{P}(TN_t|TN_{t-1})$ with the observed evidence $P_{RX,t}$. Assuming independence of the distance

Algorithm 1 Particle filtering algorithm

Initialization:
Input: $t, S_{t-1}, P_{RX,t}, \mathcal{P}$
Output: S_t
ϑ = 0, sample $S_0(s)$ from $\mathcal{P}(TN_0)$
if $t > 0$ **then**
  for $s \in$ samples **do**
    sample $S_t(s)$ from $\mathcal{P}(TN_t|TN_{t-1})$
    weight $W(s) = \mathcal{P}(P_{RX,t}|TN_t, P_{TX,t})$
    $W(s) = W(s)/\vartheta$
  end for
end if
for $s \in$ samples **do**
  $\vartheta = \vartheta + W(s)$
end for
resample $S_t(s)$ according to $W(s)$ with replacement
return S_t

Algorithm 2 Weight update

Input: $TN_t, S_t, SN_t, P_{TX,t}$
Output: $W(s)$
for $i \in$ samples **do**
  $TN_t(i) = S_t(i)$
  for $j \in SN$ **do**
    $d'_{i,j} = \|TN_i - SN_j\|$
    $\mathcal{P}(r_{i,j}|d'_{i,j}, P_{i,j}) \sim N\left(mean(d', P_{TX,t}), \sigma(d', P_{TX,t})\right)$
  end for
  $W(i) = \prod_{j \in B} \mathcal{P}(r_{i,j}|d'_{i,j}, P_{i,j})$
end for
return $W_t(i)$

measurements, the importance factors w_t^i of particles TN_t^i are determined based on a weighting function:

$$\mathcal{P}(P_{RX,t}|TN_t, P_{TX,t}) = \prod_{j \in B} \mathcal{P}(r_{ij}|TN_{ij}, P_{ij})_t = \prod_{j \in B} \mathcal{P}(r_{ij}|d'_{ij}, P_{i,j})_t \quad (25)$$

These importance factors are proportional to the likelihood of the particles given the observed evidence, reflecting their suitability as representatives of the true target position at time t.

In the resampling step, new particles are selected with a probability that is directly proportional to their weights. This means that the particles with higher weights, which correspond to the most likely representations of the target's position, have a greater chance of being chosen. Conversely, particles with lower weights, which are less likely to accurately represent the target's position, have a lower probability of being selected. Through this resampling process, the particles gradually converge toward the most likely point, step by step, resulting in a refined and more accurate estimation of the target's position (see Algorithms 1 and 2).

B. UAV Swarm Selection by e/gNB

After detecting the target TN and estimating its position, the SNs promptly transmit the target location information to the nearby e/gNB. Upon receiving these data, the e/gNB selects a swarm of UAVs to efficiently track the target TN. To facilitate this process, e/gNB employs an xApp developed on the RIC framework, enabling intelligent decision-making for resource management. Drawing on our previous work [30], the developed xApp leverages the optimized computing resource allocation (OCRA) model to allocate the most appropriate UAVs for tracking. Within this model, UAVs are treated as valuable resources, and each e/gNB maintains a pool of UAVs, selecting the optimal ones based on criteria such as power levels, proximity to the target, and available resources.

To ensure that UAVs find the most efficient path to the target, a path planning algorithm is employed (see Algorithm 3). This algorithm calculates the best flight paths considering factors such as obstacles, power efficiency, and time to the target. The selected UAVs then utilize this algorithm to determine the optimal route, ensuring they reach the target in the shortest time while conserving power.

Once the UAVs are selected, they form a cluster, with the UAV possessing the highest power level designated as the cluster head (CH). The CH plays a central role in coordinating the tracking process and managing communication among the UAVs in the cluster. By designating the UAV with the highest power level as the CH, the system ensures that the UAV is equipped with sufficient resources to handle coordination duties effectively. Algorithm 4 outlines the pseudocode for the selection process of the UAVs and the cluster head.

C. Our Multi-Agent Q-learning Algorithm

In this section, we propose our multi-agent Q-learning algorithm (MAQL) that incorporates power consumption, accuracy, delay, and avoiding collisions with obstacles in action selection. We first model the problem of target tracking by a swarm of UAVs using the Markov Decision Process (MDP). It is defined as a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R})$, where:

- $\mathbb{S}$ is the set of states $s \in \mathbb{S}$ representing the current configuration of the UAV swarm and the target.
- $\mathbb{A}$ is the set of actions $a \in \mathbb{A}$ that the UAVs can take in each state.
- $\mathbb{P}$ is the transition probability function $\mathbb{P} = \{p(s'|s,a)|s, s' \in \mathbb{S}, a \in \mathbb{A}\}$, which gives the probability of transitioning to state s' when action a is taken in state s.
- $\mathbb{R}$ is reward function, $\mathbb{R} = \{r(s,a)|s \in \mathbb{S}, a \in \mathbb{A}\}$, which assigns a numerical reward to each state-action-state' triplet.

As explained in [31], for decision problems that exhibit the Markovian property, it is possible to find an optimal policy that relies solely on the current state to

Algorithm 3 Path-planning algorithm for UAV

Input: Initial UAV position $U_{n,t}$, target position $TN_{m,t}$, obstacle positions O, UAV power level $PW_{n,t}$
Output: Optimal flight path for UAVs
Initialize UAV position: $U_{current} \leftarrow U_{n,t}$
Initialize Target position: $TN_{current} \leftarrow TN_{m,t}$
Set maximum iterations: $iter \leftarrow maxIter$
Initialize empty path: $path \leftarrow [U_{current}]$
for $i = 1$ to $iter$ **do**
  Calculate the distance to the target:
  $dist \leftarrow ||U_{current} - TN_{current}||$
  if $dist < threshold$ **then**
    Break: Target reached
  end if
  Find obstacle-free neighboring nodes from $U_{current}$
  Evaluate cost for each possible path considering:
    - Distance to the target
    - Power consumption for movement
    - Direction change penalty
  Choose the next position U_{next} with minimum cost
  Update current position: $U_{current} \leftarrow U_{next}$
  Append $U_{current}$ to $path$
  Check remaining power:
  if $PW_{n,t} < powerThreshold$ **then**
    Break: Insufficient power
  end if
  Adjust altitude if required for better RSSI or obstacle avoidance
end for
if $U_{current} \neq TN_{current}$ **then**
  Output: Path not found
else
  Output: Optimal path $path$
end if

Algorithm 4 UAV swarm & cluster head selection

Input: $Pool = \{UAV_1, UAV_2, \cdots, UAV_N\}$, $TN = (x, y)$
Output: $LU_f \subset Pool$, CH
Initialize LU_f as an empty list
$LU_f \leftarrow$ OCRA model($Pool$)
$CH \leftarrow$ Select a UAV with highest power level from LU_f
return LU_f, CH

make decisions. In the context of tracking targets by UAVs and UAV communication systems, which inherently possess the Markovian property, it is appropriate to consider Markovian policies. These policies enable UAVs to make informed decisions based on their current state without requiring extensive knowledge of past states or actions. Using Markovian policies, we can effectively optimize the tracking and communication strategies of UAVs, taking into account the dynamic nature of the environment and the evolving target movements.

The goal in solving MDP is to find an optimal policy $\pi : \mathbb{S} \rightarrow \mathbb{A}$ that maximizes the cumulative expected reward over time. This can be achieved by using reinforcement learning algorithms, such as Q-learning, to iteratively update the Q-values, $Q(s, a)$, which represent the expected cumulative reward for taking action a in state s and following the optimal policy thereafter. The optimal policy can be obtained by iteratively updating the Q-values based on Equation (5).

In the context of MDP, the expectation operator with respect to a policy π and the number of time slots T refers to the expected value of a certain quantity over a sequence of T time steps, taking into account the stochasticity of the system and the actions chosen according to the policy π. Mathematically, the expectation operator with respect to the time intervals of policy π and T is denoted as $\mathbb{E}[.|\pi, T]$ and represents the average value of a function or random variable over multiple time steps, considering the actions taken according to policy π. Considering a reward function $R(s_t, a_t)$, we can calculate the expected cumulative reward $\mathcal{R}_\pi^{(1)}$ over T time steps using the expectation operator as [32]:

$$\begin{aligned}\mathcal{Q}_\pi^{(1)} &= \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)|\pi, T\right] \\ &= \mathbb{E}\left[R(s_t, a_t) + \gamma \mathcal{Q}_\pi^{(1)}(s_{t+1}, a_{t+1})|\pi, T\right] \end{aligned} \quad (26)$$

Here, the expectation is taken over all possible sequences of state-action pairs (s_t, a_t) that can be generated by the policy π for T time steps. In our approach, a swarm of N available UAVs, where $N \geq 3$, forms a cluster to collectively track a target. As a result, Equation (26) can be reformulated as follows:

$$\mathcal{Q}_\pi^{(N)} = \arg\max_\pi \mathbb{E}\left[\sum_{n=1}^{N}\left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)|\pi, T\right]|\pi, N\right] \quad (27)$$

The objective of our proposed multi-UAV tracking approach is to maximize $\mathcal{R}_\pi^{(N)}$. The optimization problem can be formulated as follows:

$$\begin{aligned}&\max_\pi \left\{\mathcal{Q}_\pi^{(N)}\right\} \\ &\text{subject to} \quad (27)\end{aligned} \quad (28)$$

Based on the formulation of Equation (28), we will now present our MAQL algorithm. The main motivation behind our approach is to impose restrictions on agents, preventing them from taking unnecessary actions that do not align with our objectives. Our objectives include minimizing power consumption and delay, maximizing accuracy of tracking, and avoiding collisions with obstacles and other UAVs. By incorporating these restrictions, we aim to guide the decision-making process of the agents towards actions that are more aligned with our desired outcomes. In pursuit of our objectives, we devise a comprehensive reward function for UAVs, denoted by

$R_{n,t}(s,a)$, when UAV n takes action a in state s at time t as follows:

$$R_{n,t}(s,a) = r^{+}_{n,t}(s,a) + \dot{r}^{-}_{n,t}(s,a) + \ddot{r}^{-}_{n,t}(s,a) \quad (29)$$

This reward function amalgamates three crucial components, each contributing to the UAV's performance evaluation. Specifically, we express the reward function as the sum of three terms:

- $r^{+}_{n,t}(s,a)$: This term encompasses the positive rewards related to power consumption, delay, and accuracy. Reflects the efficiency of the UAV in these aspects, encouraging actions that lead to minimized power consumption, reduced delay, and improved tracking accuracy.
- $\dot{r}^{-}_{n,t}(s,a)$: This term involves the negative reward function for avoiding obstacle collisions. It penalizes actions that can potentially result in collisions with obstacles in the environment, motivating the UAV to choose paths that ensure safe navigation.
- $\ddot{r}^{-}_{n,t}(s,a)$: This term introduces the negative reward function associated with avoiding collisions with other UAVs within the cluster. It discourages actions that might lead to potential collisions with other UAVs, promoting collaborative and collision-free flight patterns.

In the following, we delve into each reward function, providing further insight into their definitions and implications for guiding the UAV's decision-making process in the dynamic and complex tracking environment.

In accordance with our objective of minimizing power consumption, delay, and maximizing tracking accuracy, we have crafted a reward function denoted as $r_{n,t}(s,a)$. This reward function is designed to encompass the essential aspects of our tracking approach, enabling us to make informed and optimal decisions. The reward function is formulated as follows:

$$r_{n,t}(s,a) = w_1 \times (1 - \mathsf{P}) + w_2 \times (1 - \mathsf{D}) + w_3 \times \mathsf{A} \quad (30)$$

where P, D, and A represent power consumption, delay, and accuracy, respectively. To achieve our objectives, we introduce weights w_1, w_2, w_3, assigned to power, delay, and accuracy.

Given the diverse nature and varying value ranges of the parameters, normalization and conversion into a common scale become essential. By normalizing these parameters, we ensure that each parameter contributes proportionately to the overall optimization process, regardless of its original scale. This normalization step enables fair comparison and effective combination of different objectives, facilitating a balanced decision-making process for our multi-objective tracking system. The normalization formula is given by [33]:

$$\tilde{Value} = \frac{Value_{src} - Value_{min}}{Value_{max} - Value_{min}} \quad (31)$$

The data normalization process involves utilizing the original data $Value_{src}$ and transforming it into a normalized value $\tilde{Value}$. The normalized value is calculated using the minimum value $Value_{min}$ and maximum value $Value_{max}$ from the original data set. In this work, we apply this method to normalize power consumption, delay, and accuracy, resulting in $\tilde{\mathsf{P}}$, $\tilde{\mathsf{D}}$, and $\tilde{\mathsf{A}}$ as normalized values for power consumption, delay, and accuracy, respectively.

In addition, in multi-objective decision making, the importance of various parameters can vary depending on different situations. The appropriate assignment of weights to these parameters is crucial to achieve optimal results. For instance, accuracy is typically prioritized over power consumption and delay, but when the UAV's power is below a certain threshold, power becomes more critical than accuracy and delay. In such dynamic scenarios, Pareto optimization proves to be a powerful approach [30]. Using Pareto optimization, the UAV system can adapt its weighting strategy based on real-time conditions, recalculating the optimal trade-offs and solutions [34]. Implementing dynamic weighting with Pareto optimization involves the following steps. Firstly, create a weighting policy that defines how the weights should be adjusted based on environmental and mission-specific factors. Continuously monitor the environment to identify changes or specific scenarios that require different weightings. Once detected, update the weights based on information from the environment and the established policy. Subsequently, recalculate the Pareto frontiers using the new weights to identify updated trade-offs and optimal solutions that align with the evolving priorities. This approach allows the UAV cluster to respond effectively to changing conditions, making informed decisions and optimizing its performance in dynamic and challenging environments. Therefore, based on normalization, we reformulate Equation (30) as follows:

$$r^{+}_{n,t}(s,a) = w_1 \times \left(1 - \tilde{\mathsf{P}}\right) + w_2 \times \left(1 - \tilde{\mathsf{D}}\right) + w_3 \times \tilde{\mathsf{A}} \quad (32)$$

In addition to these objectives, our approach also takes into account the need to avoid collisions with obstacles in the urban environment. This is incorporated into the reward function as a negative reward. As detailed in Section A, when each UAV enters the e/gNB area, it quickly acquires the necessary information on the obstacles present within that specific area from the e/gNB. This exchange of information ensures that the UAV is well informed about the obstacles in its vicinity, allowing it to navigate safely and effectively during the tracking process. Let the environment be represented by a 3-D grid of size $\mathcal{X} \times \mathcal{Y} \times \mathcal{H}$, where each cell is denoted by (x, y, z), with $1 \leq x \leq \mathcal{X}$, $1 \leq y \leq \mathcal{Y}$, and $1 \leq z \leq \mathcal{H}$. In this environment, when an obstacle is present at coordinates (x, y, z), we represent the corresponding cell on the grid as 1, denoted as $\mathcal{O}(x, y, z) = 1$, signifying the location of the obstacle. In contrast, if the cell is obstacle-free, we set $\mathcal{O}(x, y, z) = 0$, which indicates an unobstructed area.

When a UAV is located at position $(\dot{x}, \dot{y}, \dot{z})$ and takes action a, it results in a move to a new location $(\dot{x}' = \dot{x} \pm \Delta x_a, \dot{y}' = \dot{y} \pm \Delta y_a, \dot{z}' = \dot{z} \pm \Delta z_a)$. Here, Δx_a, Δy_a, and Δz_a represent the respective displacements of the UAV on the x-axis, y-axis, and z-axis according to the selection of the action a. To avoid collisions with

obstacles, it needs UAVs to compute the probability of collision using a probabilistic approach based on each action. Let $P_o(a) \in [0,1]$ be the probability of an obstacle collision for the action a. Since UAVs know the locations of obstacles, they can directly check if any obstacle lies in the path of action a and calculate the probability accordingly. If any cell with $\mathcal{O}(x,y,z) = 1$ and $z \geq \dot{z}$ lies between $(\dot{x},\dot{y},\dot{z})$ and $(\dot{x}',\dot{y}',\dot{z}')$ resulting from taking action a, then the probability of collision $P_o(a)$ is set to 1, indicating a high probability of collision. However, if there are obstacles with $\mathcal{O}(x,y,z) = 1$ along the action trajectory a, but these obstacles are encountered after reaching the new location, the UAV measures the distance $Dist_{u'o}$ between the last known position $(\dot{x}',\dot{y}',\dot{z}')$ and the obstacle location (x_o,y_o,z_o) along the trajectory of action a. In addition, it calculates the distance $Dist_{uo}$ between the current location of the UAV $(\dot{x},\dot{y},\dot{z})$ and the obstacle location (x_o,y_o,z_o). Based on these measurements, the probability of collision $P_o(a)$ is calculated as follows:

$$P_o(a) = \frac{Dist_{uo} - Dist_{u'o}}{Dist_{uo}} \tag{33}$$

On the other hand, if there is no risk of collision, meaning $\mathcal{O}(x,y,z)$ for all cells in the path is 0, then $P_o(a)$ is set to 0.

In such cases, we include a negative reward $\dot{r}^-_{n,t}(s,a)$ in the overall reward function as follows:

$$\dot{r}^-_{n,t}(s,a) = P_o(a) \times -r^+_{n,t}(s,a) \tag{34}$$

This negative reward represents the penalty incurred for the possibility of colliding with an obstacle during the action. If there is no potential for collision with obstacles along the trajectory of action a, the UAV does not receive any negative reward and the overall reward is unaffected. However, if there is a possibility of collision, the negative reward is incorporated to discourage the UAV from selecting actions during its flight that could lead to collisions with obstacles.

In addition, since a group of UAVs fly closely together to track a target, it is crucial that each UAV avoid collisions with other UAVs within the group during target tracking. To address this issue, we have two options: (i) By forcing UAVs to fly at different altitudes, we can reduce the risk of collisions between them. This approach ensures that each UAV maintains a safe distance from other UAVs within the swarm, minimizing the chances of interference and accidents. However, it is important to note that this approach may result in an increase in the power consumption of UAVs that fly at higher altitudes. (ii) Introduce an additional negative reward to our approach denoted as $\ddot{r}^-_{n,t}(s,a)$. This negative reward is applied when the UAV n in state s takes action a at time t, and this action increases the probability of collision with other UAVs in the swarm. In contrast to fixed obstacles, the dynamic nature of mobile obstacles, such as other UAVs, introduces unpredictability into the environment. Hence, it becomes crucial to account for probabilities when avoiding collisions with these moving obstacles.

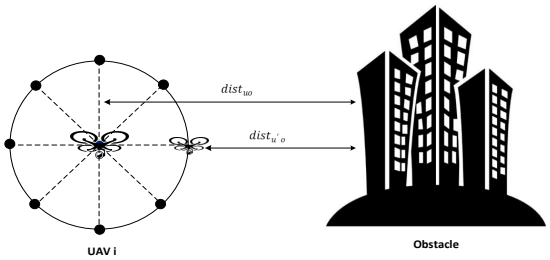


Figure 2: Distance between UAV and an obstacle.

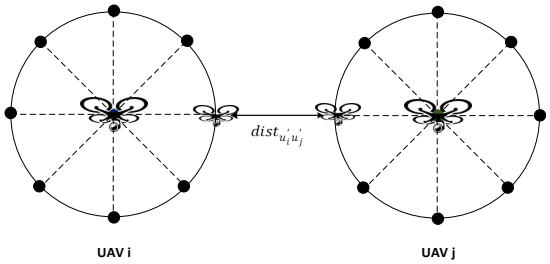


Figure 3: Distance between two UAVs after selecting the next action.

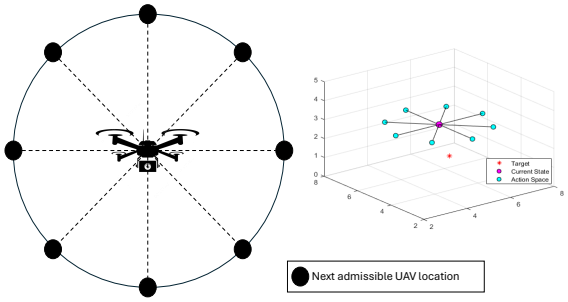


Figure 4: All allowable actions at each state s.

Let $P_u(a) \in [0,1]$ represent the probability of collision with other UAVs in the group for action a. Since each UAV in the cluster has knowledge of the last locations of other UAVs and the location of the target node, it can compute the probability $P_u(a)$ for each action based on this information. To do this, each UAV will check if there is any overlap between its planned trajectory and the trajectory of the other UAV. This is accomplished by calculating the Euclidean distance $\|.\|$ between two UAVs i and j, known as $Dist_{u_i u_j}$. Furthermore, the UAV i measures the distance between its current location $(\dot{x}_i,\dot{y}_i,\dot{z}_i)$ and the new location after taking action a $(\dot{x}'_i,\dot{y}'_i,\dot{z}'_i)$, denoted $Dist_{u_i u'_i}$. The UAV i also performs the same calculations for all the actions that UAV j may take and lead to two UAVs approaching each other. Next, the UAV i computes the distance between $(\dot{x}'_i,\dot{y}'_i,\dot{z}'_i)$ and $(\dot{x}'_j,\dot{y}'_j,\dot{z}'_j)$, denoted as $Dist_{u'_i u'_j}$. This evaluation allows each UAV to anticipate the movements of its neighboring UAVs and identify any potential overlaps between their respective planned trajectories. We also defined a certain threshold value, which represents the minimum safe dis-

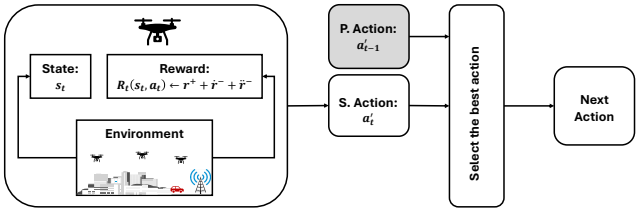


Figure 5: eMAQL.

tance $Dist_s$ to avoid collision. We compute $P_u(a)$ based on the $Dist_{u'_i u'_j}$ and $Dist_s$ as follows:

$$P_u(a) = 1 - \frac{Dist_{u'_i u'_j}}{Dist_s} \tag{35}$$

This probability indicates the likelihood that the selected action may lead to a collision with other UAVs within the tracking cluster. If $P_u(a)$ equals 1, indicating a high likelihood of collision, otherwise, if $P_u(a)$ equal to 0, there is no risk of collision.

In the next step, we compute the negative reward $\ddot{r}^-_{n,t}(s,a)$ as follows and include it in the overall reward function. This integration emphasizes UAV decision making that actively avoids collisions with other UAVs within the cluster, effectively improving safety and target tracking efficiency.

$$\ddot{r}^-_{n,t}(s,a) = P_u(a) \times -r^+_{n,t}(s,a). \tag{36}$$

By including these reward components, our approach encourages UAVs to make decisions that prioritize collision avoidance while still optimizing power, delay, and accuracy objectives. In our MAQL algorithm, during the target tracking process, each UAV in each state s is required to employ these reward components and measure the total reward value $R(s,a)$ for all admissible actions $a \in \mathring{A} = \left\{ \frac{k\pi}{180} \mid \forall\, k \in N, 1 \leq k \leq 360 \right\}$. These admissible actions represent various flight directions along the x, y, and z axes. Individually, every UAV manages its exclusive Q-table, which includes both the $R(s,a)$ values and the corresponding Q-values $Q(s_t, a_t)$ for all allowable actions in each state s. The Q-table is continuously updated as the UAV transitions between states during the target tracking process. By storing measured Q-values, each UAV can make informed decisions about its next actions, selecting the actions that maximize the expected cumulative reward, as defined in Equation (28). This enables the UAV to efficiently track the target, navigate the environment effectively, and optimize its trajectory for improved tracking performance.

D. Enhanced-MAQL (eMAQL)

In our MAQL, UAVs monitor the current state of the target and determine the next action based solely on the current scenario. Although this method allows real-time adaptability, it can result in inefficiencies, particularly in environments where the target's movement is unpredictable or erratic [35]. Specifically, since targets can change direction frequently, UAVs need to adjust their flight path accordingly, often leading to a zigzag movement pattern. This pattern not only results in inefficient tracking but also causes a significant increase in UAV energy consumption as a result of constant course corrections.

To address this issue, we propose eMAQL, which extends MAQL by incorporating past actions into the state representation, effectively maintaining a form of memory. Unlike MAQL, where the problem is modeled as a Markov Decision Process (MDP), assuming that the future state depends solely on the current state and action, real-world UAV tracking exhibits the characteristics of a Partially Observable Markov Decision Process (POMDP) [36], [37]. This is because UAVs receive partial and noisy observations of the target's position due to factors such as sensor noise, environmental occlusions, and unpredictable target maneuvers. In practical UAV tracking scenarios, relying solely on the current state without incorporating historical data can lead to suboptimal decision-making, particularly when observations are unreliable or incomplete. By integrating past actions into the state representation, eMAQL enables UAVs to leverage historical context, improving their ability to infer hidden states, predict future movements, and make more informed navigation decisions. This memory-enhanced approach helps mitigate the uncertainties introduced by partial observability, leading to more stable and accurate target tracking in dynamic environments.

Algorithm 5 UAV replacement

Initialization:
$PW^{(w)}$: Low power threshold
$PW^{(e)}$: Critical power threshold
$Pool$: A Pool of available UAVs
CH: Cluster Head
e/gNB: Base Station
while Target tracking is ongoing **do**
  for Each UAV in the cluster **do**
    if $PW_{U,t} < PW^{(w)}$ **then**
      UAV sends a warning message to CH and e/gNB
      UAV turns off non-essential functions
      UAV receives target location from CH
    else if $PW_{U,t} < PW^{(e)}$ **then**
      UAV sends an error message to CH and e/gNB
      UAV initiates landing procedure
    end if
  end for
  if Warning message received by e/gNB **then**
    e/gNB finds a new UAV from its $Pool$
    Dispatch the new UAV to replace the warning-sending UAV
  end if
end while

As illustrated in Figure 5, eMAQL enhances the decision-making process by determining the next optimal move for UAVs. Specifically, the UAV selects the best action from the set of possible actions $a_t^{'}$, based on the current state s_t, the current action a_t, and the previous action $a_{t-1}^{'}$. This enhanced decision-making process allows UAVs to make more informed choices about their next action, considering not only the present situation, but also their previous movements.

By factoring in the last action, eMAQL introduces a form of movement prediction that mitigates unnecessary back-and-forth movement patterns, leading to a smoother trajectory. This results in reduced power consumption, as fewer course corrections are needed, and better overall tracking performance. In addition, eMAQL helps UAVs maintain better spatial awareness, allowing more strategic tracking of mobile targets, even in complex environments. The benefits of eMAQL can be summarized as follows:

- Reduction in zigzag movements: Considering previous actions, eMAQL allows UAVs to move more smoothly and predictably, avoiding inefficient zigzag patterns.
- Improved power efficiency: The smoother movement pattern significantly reduces the power used in frequent direction changes, extending the operational time of UAVs.
- Increased tracking accuracy: As UAVs are better able to predict and adjust to the target's movements, the overall accuracy of target tracking is improved.
- Better adaptation to dynamic environments: eMAQL enables UAVs to adapt more effectively to dynamic environments with constantly changing target movements, thereby enhancing overall system performance.

E. Cluster Dynamics and Communication Protocols

In this work, we employ a swarm of UAVs organized into clusters, each led by a CH. Throughout the target tracking process, robust communication channels are established between individual UAVs and their respective CHs. For example, as the power of a UAV decreases to a level lower than a predefined threshold, denoted $PW^{(w)}$, the UAV autonomously triggers a warning signal directed towards the nearby e/gNB and its CH. This warning signifies that the UAV has reached a critical power state and requires replacement with a fresh UAV. Consequently, the UAV optimizes its operations by deactivating nonessential functions, subsequently relying on the CH to relay target location information. Upon receiving a warning message from a UAV, the EN assumes the responsibility of identifying a replacement UAV from its available pool and dispatching the newly assigned UAV to take the place of the one that triggered the warning. Furthermore, should a UAV's power level drop below a critical threshold represented as $PW^{(e)}$, the UAV will promptly transmit an error message to both its CH and the neighboring EN. This signal indicates that the UAV has depleted its power reserves to a point where it can no longer support the tracking operation and must initiate a controlled landing procedure. Upon arrival or departure of a UAV in or out of the cluster, the CH promptly disseminates a notification message to inform the remaining UAVs within the cluster. This message serves to keep all UAVs within the cluster aware of changes in their group composition, ensuring seamless coordination during target tracking operations. This process ensures uninterrupted target tracking while efficiently managing the UAV fleet's power. Algorithm 5 outlines the replacement process of UAVs when their power levels fall below critical thresholds, ensuring uninterrupted target tracking while efficiently managing the power resources of the UAV fleet.

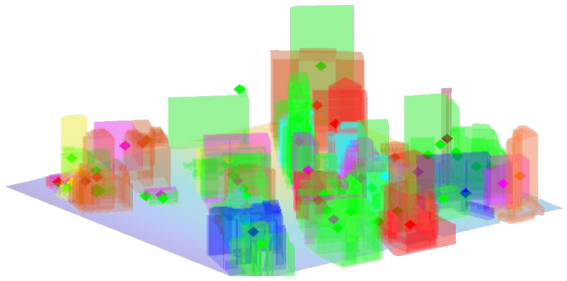


Figure 6: 3D urban environment with buildings as obstacles.

Table I: Simulation setup parameters

Parameter	Value
Number of UAVs	3 (Black - Blue - Green)
Number of e/gNBs	2
Number of SNs	10 - 500
Target Start Coordinates	(10,60)
Target End Coordinates	(300,150)
Number of Obstacles	100
UAV Speed Range	0 - 5 m/s
Target Average Velocity	3 m/s (Red)
Action Space for UAVs	8
UAV Mass	4kg

VI. Numerical Experiments

In this work, our goal is to reduce the power consumption and delay while improving the accuracy and overall performance of our tracking system. To this end, we compute several important factors: Root Mean Square Error (RMSE), measured in meters, quantifies tracking accuracy by evaluating the precision of the estimated target positions; power consumption, quantifying the power used by UAVs during tracking; communication latency, assessing the communication time between UAVs and the e/gNBs; and collision avoidance, evaluating the effectiveness of our strategy in preventing collisions with obstacles and other UAVs. We also present the results of our simulations and discuss the performance of PF, enhanced PF (ePF), MAQL, and eMAQL.

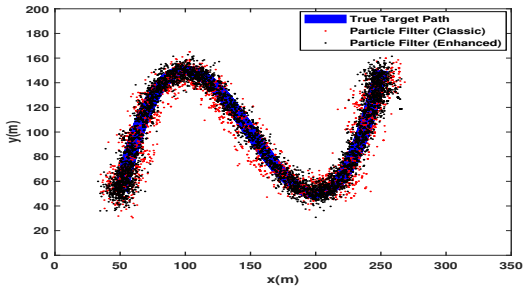


Figure 7: Particle history: PF vs ePF

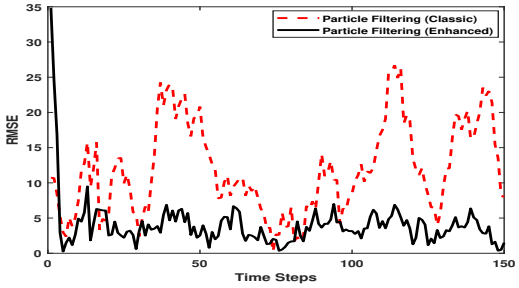


Figure 8: RMSE comparison: PF vs ePF

A. Simulation Setup

In this section, we outline the simulation setup used to assess the performance of our proposed algorithms: PF, MAQL, and eMAQL. The simulations were carried out using MATLAB and Simulink on a laptop with an 11th Gen Intel Core i7-1165G7 processor and 16GB of RAM. We modeled a 3D urban environment with randomly placed targets and obstacles, including buildings, streets, and open areas, as shown in Figure 6. The targets followed different paths generated by a path planning algorithm to avoid collisions with obstacles, which included fixed structures (buildings) and dynamic ones (such as other UAVs). The system includes a single target, represented in Red, which is tracked by multiple UAVs shown in Black, Blue, and Green, along with two e/gNBs and 10-500 SNs deployed in an environment containing 100 cylindrical obstacles. The UAVs operate at speeds ranging from 0 to 5 m/s, while the target moves along a trajectory from coordinates (10,60) to (300,150) at an average speed of 3 m/s. Each UAV weighs 4 kg, and the eMAQL algorithm uses an action space of $\{4, 8, 12, 36\}$ distinct actions. Table I details the simulation parameters.

B. Analysis the PF-based Localization Algorithm

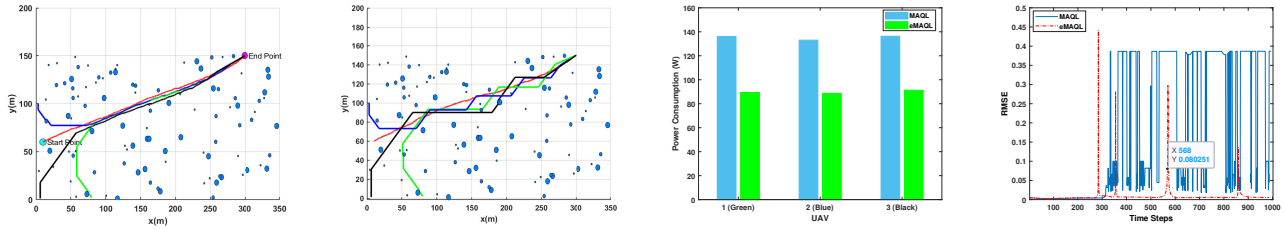
Here, we implemented a classical PF algorithm and compared it with the ePF algorithm in terms of RMSE to evaluate their effectiveness in target localization. The key difference between classical PF and ePF lies in the number of particles, noise handling, and resampling strategy. ePF uses more particles, reduces noise, and applies a smarter resampling threshold. The results obtained illustrated in Figures 7 and 8 show that the ePF significantly outperforms the classical PF, achieving a lower RMSE in various scenarios. This improvement can be attributed to the increase in the number of particles, reduced measurement noise, and a more sophisticated resampling strategy used in ePF. Consequently, the enhanced algorithm provides a more accurate and reliable estimate of the target's position, demonstrating its suitability for applications requiring precise tracking in dynamic environments.

C. Analysis and Comparison MAQL and eMAQL

In this section, we analyze and compare the performance of the MAQL and eMAQL algorithms implemented in terms of UAV trajectories, tracking accuracy (measured by RMSE) and power consumption. As shown in Figures 9a and 9b, the use of eMAQL significantly reduces the number of direction changes, or zigzag movements, made by UAVs. This reduction has a direct impact on power consumption, as highlighted in Figure 9c, where the power consumed by UAVs drops to around 60W when using eMAQL, compared to the higher consumption with MAQL. Furthermore, Figure 9d illustrates that the zigzag movement also affects the RMSE, which reflects the accuracy of the target tracking. Although the maximum RMSE for both MAQL and eMAQL is approximately 0.45, which is relatively low, eMAQL demonstrates improved performance in terms of accuracy. The improved performance of eMAQL is due to its ability to optimize decision-making using historical trajectory adjustments, reducing unnecessary repositioning, and improving movement efficiency. As explained in Section V.D, eMAQL inherently aligns with the POMDP framework and accounts for past actions, leading to smoother UAV motion, more accurate tracking, and lower energy consumption. These enhancements ensure that UAVs maintain an optimal balance between tracking accuracy and power efficiency, making eMAQL a more effective solution for real-world UAV tracking applications.

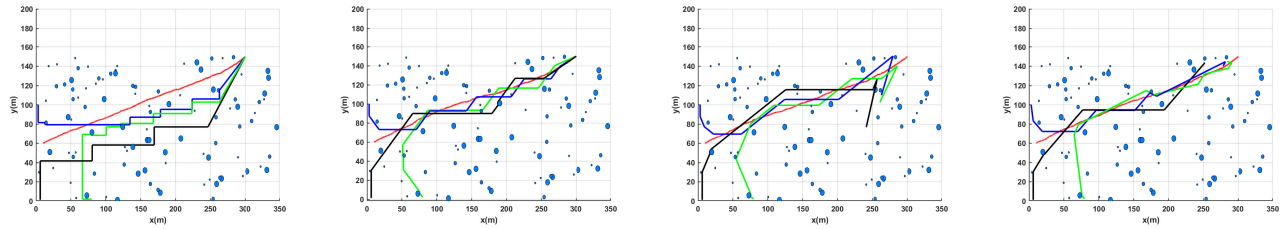
D. Analysis eMAQL with Different Number of Actions

In this section, we analyze the performance of eMAQL by using different action spaces: $\{4, 8, 12, 36\}$ actions. Figure 10 presents the UAV trajectories for each action space. It is clear that the accuracy of the eMAQL tracking improves as the number of actions increases from 4 to 8, 12, and 36. However, there is minimal difference in performance between eMAQL with 8, 12, and 36 actions. The UAVs' directional changes are also similar when using these higher numbers of actions, with little impact on the overall movement patterns. Additionally, the figure shows that eMAQL performs well in avoiding collisions, as the UAVs successfully navigate around obstacles without hitting them.



(a) Trajectory - MAQL (b) Trajectory - eMAQL (c) Power consumption (d) RMSE

Figure 9: Comparison of MAQL and eMAQL in terms of trajectory, power consumption, and RMSE.



(a) 4 Actions (b) 8 Actions (c) 12 Actions (d) 36 Actions

Figure 10: Target and UAV trajectories using eMAQL with 4, 8, 12, and 36 actions.

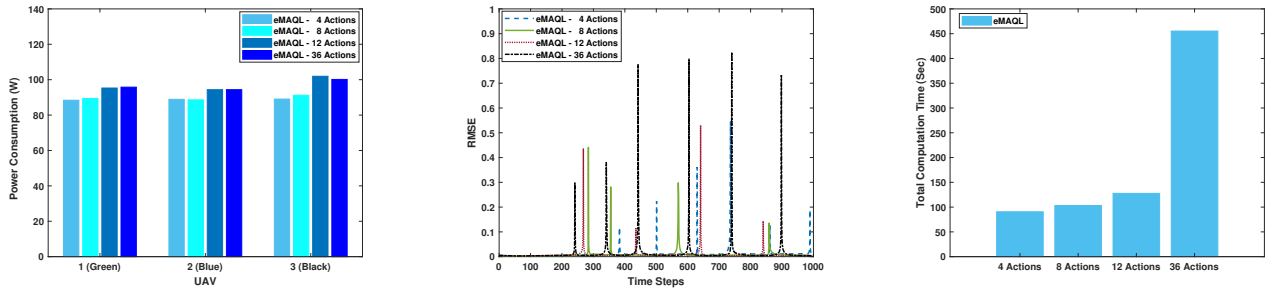
Furthermore, Figure 11a compares the power consumption of the UAVs for different action spaces. We observe that power consumption for 4 and 8 actions is nearly the same, while the use of 12 or 36 actions results in a slight increase of approximately 10W. This increase occurs because a larger action space requires more frequent decision-making and adjustments, leading to higher energy usage. However, the difference in power consumption remains small, making the additional computational complexity of higher action spaces unnecessary. In terms of accuracy, as shown in Figure 11b, we observed that the performance remains quite similar in 4, 8, 12, and 36 actions, with no significant improvement beyond 8 actions. When it comes to the computation time to track the target during its movement, as seen in Figure 11c, we find that it remains around 100 seconds for 4, 8, and 12 actions. However, it increases significantly to 455.29 seconds when using 36 actions in the eMAQL algorithm. Therefore, after considering the accuracy, power consumption, and computational efficiency tracking, we conclude that using eMAQL with 8 actions strikes the best balance between performance and resource usage.

E. Analysis The Impact of Open RAN on Target Tracking

In this section, we analyze the impact of RIC in the Open RAN environment on UAV-based target tracking. The open RAN network offers significant advantages over traditional cloud and edge computing when it comes to reducing latency in real-time applications, such as UAV-based target tracking. In cloud computing, UAVs must transmit their data to a centralized server, where computations such as target localization and trajectory planning are performed. Although cloud computing provides high processing power, it introduces significant delays due to long-distance data transmission. Edge computing aims to reduce this delay by processing data closer to the source. In edge computing, computations occur at the network edge, such as on edge servers located near UAVs. Although this architecture reduces latency compared to cloud computing, it still involves intermediate processing steps, resulting in higher communication and computation delays than Open RAN. In cloud and edge computing, communication between UAVs and cloud or edge servers in an LTE/5G network is established through e/gNBs.

In the Open RAN environment, key decision-making processes are decentralized, and components such as the non/near-RT RIC and the r/xApps deployed in these components are integrated into the RAN itself, allowing the system to process data closer to the source with minimal delay.By eliminating the need for intermediate nodes and central servers, Open RAN achieves much lower latency than cloud or edge computing architectures. This reduction in communication delay, which can be as low as 10-20 milliseconds, makes Open RAN ideal for applications where real-time responsiveness is critical, such as UAV coordination and target tracking. In our simulations, Open RAN allowed UAVs to adjust their trajectories almost instantaneously, significantly improving the accuracy and performance of the tracking system.

As shown in Figure 12, the results obtained clearly highlight the performance advantages of Open RAN over edge and cloud computing in terms of reducing latency



(a) Power Consumption Comparison (b) RMSE Comparison (c) Total Computation Time

Figure 11: Comparison eMAQL with different actions 4, 8, 12, and 36.

and improving the accuracy of target tracking. The simulation demonstrates that the Open RAN-based system consistently outperforms the cloud and edge/fog systems, especially as the number of UAVs increases. RMSE values are significantly lower in Open RAN, indicating better accuracy in estimating the target position. Furthermore, the communication delay in the Open RAN system is nearly half that of the edge system and dramatically less than that of the cloud-based approach. These results justify our choice of Open RAN as the preferred architecture for this UAV-based target tracking system, proving its ability to efficiently support real-time operations.

F. Analysis the Impact of Task Offloading

Task offloading plays an important role in reducing the power consumption of UAVs. In a no-offloading scenario, UAVs need to handle all computational tasks on their own, which leads to higher power usage. For example, in our model with eMAQL using 8 actions, each UAV consumes about 90 watts during the entire tracking process. This power consumption results from the UAVs running algorithms, processing data, and staying in constant communication with e/gNBs, all of which heavily use the UAVs' limited onboard resources.

On the other hand, partial offloading allows UAVs to send some of their computational tasks to the e/gNBs, reducing their local workload and saving power. In this case, only half of the tasks are done locally, lowering the power consumption to 40 watts, plus an additional 10 watts for managing communication. When all tasks are fully offloaded to e/gNB, the UAVs only need to handle communication, resulting in a much lower power consumption of around 15 watts per UAV (see Figure 13). This shows that offloading tasks, whether partially or fully, can greatly decrease the power used by UAVs, making them more energy efficient for longer missions.

Task offloading also affects the accuracy of the tracking, mainly due to communication delays between UAVs and e/gNBs. As UAVs rely on external computation, delays in processing and returning results can lead to outdated target position estimates, reducing tracking precision. As shown in Figure 13, partial offloading maintains a good balance between energy efficiency and tracking accuracy, with only a minor increase in RMSE. In fully offloaded scenarios, where all computational tasks are processed externally, RMSE increases significantly as UAVs receive delayed updates, making it harder to precisely adjust their flight paths in real time. On the other hand, while RMSE in the no-offloading scenario is lower compared to partial and full offloading, it results in faster power depletion, requiring UAV replacement, which may disrupt target tracking.

G. Comparative Analysis of eMAQL, CRLB, and DRL

In this section, we compare the performance of the eMAQL with eight actions, the Cramér-Rao Lower Bound (CRLB) and Deep Reinforcement Learning (DRL) algorithms in terms of RMSE for target tracking. The CRLB is a theoretical lower bound on the variance of an unbiased estimator that defines the best possible accuracy that any estimator can achieve. The DRL utilizes a deep Q-network to handle a continuous state space with a discrete action set while incorporating a particle filter to estimate multiple target states. The eMAQL algorithm, as discussed earlier, has demonstrated strong performance in tracking accuracy due to its ability to efficiently manage the UAV's action space and minimize unnecessary movements like zigzag patterns. DRL has also shown potential in tracking tasks, but its performance lags slightly behind eMAQL due to the higher complexity in decision making and training.

Figure 14 illustrates the RMSE comparison between CRLB, eMAQL with 8 actions, and DRL over time. As expected, CRLB achieves the lowest RMSE, indicating the best tracking accuracy in all time steps. eMAQL follows closely, with RMSE values consistently approaching the CRLB results, proving that our algorithm efficiently tracks the target with minimal error. On the other hand, DRL shows relatively higher RMSE values compared to both CRLB and eMAQL, but remains close to our proposed eMAQL model. The difference in performance between DRL and eMAQL comes from DRL's time-consuming training process and complex state-space representations, which make real-time UAV adjustments

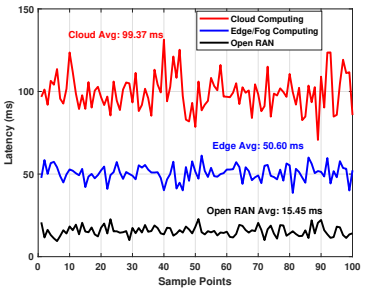


Figure 12: Open-RAN vs Edge/Cloud.

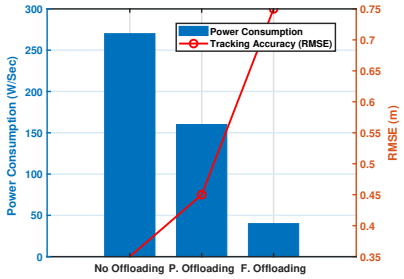


Figure 13: All task offloading strategies.

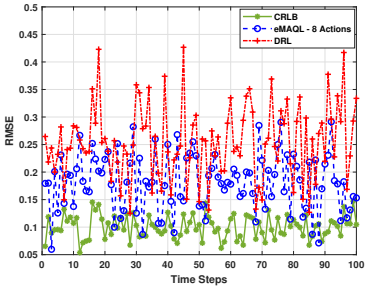


Figure 14: CRLB vs eMAQL vs DRL.

more difficult. DRL relies on deep neural networks that require extensive training and continuous weight updates, making learning much slower than eMAQL. Additionally, DRL processes high-dimensional state representations, increasing computational complexity and decision delays. In contrast, eMAQL efficiently updates values with simpler state representations, allowing UAVs to adapt faster and achieve better tracking performance in dynamic environments.

The comparison confirms that eMAQL with 8 actions achieves a balance between computational efficiency and tracking accuracy, making it highly suitable for practical applications in dynamic environments with limited resources. DRL, while effective, has higher computational demands and is less efficient than eMAQL in real-time tracking scenarios. Based on this analysis, eMAQL proves to be an effective and near-optimal solution for UAV-based target tracking.

H. Impact of Varying Reference Power Levels (PL_0)

Here, we analyze the effect of varying the PL_0 on the localization of the target. As shown in Figure 15, assuming a fixed PL_0 leads to an RMSE of 1.44 meters, while incorporating real-world variations in PL_0 increases the RMSE to 1.63 meters. This outcome, a relatively small difference in RMSE, indicates that the fixed assumption PL_0 used in our work is a reasonable approximation, as it is not far from the range of variations observed under real-world conditions. For this study, we set $PL_0 = 61$ dB and $P_{TX} = 33$ dBm, based on realistic vehicular communication settings. The reference power level at 1 meter,$PL_0 = 20\log_{10}\left(\frac{4\pi f d}{c}\right)$, is derived from the 28 GHz free-space path loss model and the speed of light c, which are widely used in vehicular and urban communication systems. P_{TX} is also aligned with standard values for Vehicle-to-Everything (V2X) communication, ensuring a realistic representation of RF signal behavior in dense urban environments.

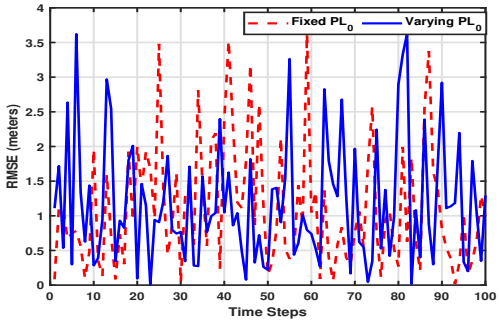


Figure 15: Fixed PL_0 vs varying PL_0

VII. Conclusion

This study has presented a robust framework for tracking targets in complex urban environments, successfully demonstrating the effective integration of mobile sensor nodes and UAVs within an Open-RAN architecture. The findings reveal that the eMAQL algorithm optimally operates with 8 actions, achieving an average power consumption of approximately 90 watts while maintaining RMSE to estimate the target position below 0.5 m. This also highlighted the potential of our approach to effectively balance power efficiency and tracking accuracy. Future work will focus on further refining the eMAQL algorithm to enhance its adaptability to dynamic urban conditions and exploring the incorporation of real-time data from additional sensor modalities to improve tracking reliability. In addition, our objective is to investigate the scalability of the proposed framework in larger urban settings and its applicability to other applications, such as disaster response and search and rescue missions. We provide public access to the code here: https://github.com/ahmadsoleymani/UAV-target-tracking.git.

References

[1] X. Chen, Z. Feng, Z. Wei, F. Gao, and X. Yuan, "Performance of joint sensing-communication cooperative sensing UAV network," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 545–15 556, 2020.

[2] H. Oh, S. Kim, H.-s. Shin, and A. Tsourdos, "Coordinated standoff tracking of moving target groups using multiple UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1501–1514, 2015.

[3] J. Moon, S. Papaioannou, C. Laoudias, P. Kolios, and S. Kim, "Deep reinforcement learning multi-UAV trajectory control for target tracking," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15 441–15 455, 2021.

[4] A. Guerra, D. Dardari, and P. M. Djuric, "Dynamic radar networks of UAVs: A tutorial overview and tracking performance comparison with terrestrial radar networks," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 113–120, 2020.

[5] D. Huo, L. Dai, R. Chai, R. Xue, and Y. Xia, "Collision-free model predictive trajectory tracking control for UAVs in obstacle environment," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 3, pp. 2920–2932, 2022.

[6] X. Jiang, N. Li, Y. Guo, D. Yu, and S. Yang, "Localization of multiple rf sources based on bayesian compressive sensing using a limited number of UAVs with airborne RSS sensor," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 7067–7079, 2020.

[7] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, no. 1, pp. 109–137, 2023.

[8] K. Meng, Q. Wu, J. Xu, W. Chen, Z. Feng, R. Schober, and A. L. Swindlehurst, "UAV-enabled integrated sensing and communication: Opportunities and challenges," *IEEE Wireless Communications*, 2023.

[9] D. Yuan, X. Chang, Z. Li, and Z. He, "Learning adaptive spatial-temporal context-aware correlation filters for UAV tracking," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 18, no. 3, pp. 1–18, 2022.

[10] M. Park, S. An, J. Seo, and H. Oh, "Autonomous source search for UAVs using gaussian mixture model-based infotaxis: Algorithm and flight experiments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4238–4254, 2021.

[11] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2019.

[12] J. Yu, X. Liu, Y. Gao, and X. Shen, "3D channel tracking for UAV-satellite communications in space-air-ground integrated networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 12, pp. 2810–2823, 2020.

[13] Y. Lun, H. Wang, J. Wu, Y. Liu, and Y. Wang, "Target search in dynamic environments with multiple solar-powered UAVs," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9309–9321, 2022.

[14] H. Wu, H. Li, R. Xiao, and J. Liu, "Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm," *Physica A: Statistical Mechanics and its Applications*, vol. 491, pp. 127–141, 2018.

[15] Y.-J. Chen, D.-K. Chang, and C. Zhang, "Autonomous tracking using a swarm of UAVs: A constrained multi-agent reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 702–13 717, 2020.

[16] S. Li, X. Yang, X. Wang, D. Zeng, H. Ye, and Q. Zhao, "Learning target-aware vision transformers for real-time UAV tracking," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[17] S. A. Soleymani, M. Shojafar, C. H. Foh, S. Goudarzi, and W. Wang, "Secure target-tracking by UAVs in O-RAN environment," in *2024 IFIP Networking Conference (IFIP Networking)*. IEEE, 2024, pp. 204–212.

[18] H. Sallouha, M. M. Azari, A. Chiumento, and S. Pollin, "Aerial anchors positioning for reliable RSS-based outdoor localization in urban environments," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 376–379, 2017.

[19] X. Li, "RSS-based location estimation with unknown pathloss model," *IEEE Transactions on Wireless Communications*, vol. 5, no. 12, pp. 3626–3633, 2006.

[20] S. Papaioannou, S. Kim, C. Laoudias, P. Kolios, S. Kim, T. Theocharides, C. Panayiotou, and M. Polycarpou, "Coordinated CRLB-based control for tracking multiple first responders in 3D environments," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1475–1484.

[21] N. Zhou, D. Meng, and S. Lu, "Estimation of the dynamic states of synchronous machines using an extended particle filter," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4152–4161, 2013.

[22] S. S. Dias and M. G. Bruno, "Cooperative target tracking using decentralized particle filtering and RSS sensors," *IEEE Transactions on Signal Processing*, vol. 61, no. 14, pp. 3632–3646, 2013.

[23] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Learning for dynamics and control*. PMLR, 2020, pp. 486–489.

[24] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *Robotica*, vol. 17, no. 2, pp. 229–235, 1999.

[25] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5505–5516, 2019.

[26] X. Gu, G. Zhang, M. Wang, W. Duan, M. Wen, and P.-H. Ho, "UAV-aided energy-efficient edge computing networks: Security offloading optimization," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4245–4258, 2021.

[27] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Empirical power consumption model for UAVs," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[28] H. Ren and M. Q.-H. Meng, "Power adaptive localization algorithm for wireless sensor networks using particle filter," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 5, pp. 2498–2508, 2008.

[29] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[30] S. Goudarzi, S. A. Soleymani, W. Wang, and P. Xiao, "UAV-enabled mobile edge computing for resource allocation using cooperative evolutionary computation," *IEEE Transactions on Aerospace and Electronic Systems*, 2023.

[31] Y. Lai, Y. L. Che, S. Luo, and K. Wu, "Optimal wireless information and energy transmissions for UAV-enabled cognitive communication systems," in *2018 IEEE International Conference on Communication Systems (ICCS)*. IEEE, 2018, pp. 168–172.

[32] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7957–7969, 2019.

[33] H. Sun, B. Zhang, X. Zhang, Y. Yu, K. Sha, and W. Shi, "FlexEdge: Dynamic task scheduling for a UAV-based on-demand mobile edge server," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 983–16 005, 2022.

[34] G. An, Z. Wu, Z. Shen, K. Shang, and H. Ishibuchi, "Evolutionary multi-objective deep reinforcement learning for autonomous UAV navigation in large-scale complex environments," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 633–641.

[35] J. Zhang, Z. Shi, A. Zhang, Q. Yang, G. Shi, and Y. Wu, "UAV trajectory prediction based on flight state recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 60, no. 3, pp. 2629–2641, 2023.

[36] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[37] Y. Xue and W. Chen, "A UAV navigation approach based on deep reinforcement learning in large cluttered 3D environments," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3001–3014, 2022.