Article

Multiple Unmanned Aerial Vehicle (multi-UAV) Reconnaissance and Search with Limited Communication Range Using Semantic Episodic Memory in Reinforcement Learning

Boquan Zhang *, Tao Wang, Mingxuan Li, Yanru Cui, Xiang Lin and Zhi Zhu

College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; wangtao1976@nudt.edu.cn (T.W.); limingxuan23@nudt.edu.cn (M.L.); cuiyanru@nudt.edu.cn (Y.C.)
* Correspondence: zhangboquan18@alumni.nudt.edu.cn

Abstract: Unmanned Aerial Vehicles (UAVs) have garnered widespread attention in reconnaissance and search operations due to their low cost and high flexibility. However, when multiple UAVs (multi-UAV) collaborate on these tasks, a limited communication range can restrict their efficiency. This paper investigates the problem of multi-UAV collaborative reconnaissance and search for static targets with a limited communication range (MCRS-LCR). To address communication limitations, we designed a communication and information fusion model based on belief maps and modeled MCRS-LCR as a multi-objective optimization problem. We further reformulated this problem as a decentralized partially observable Markov decision process (Dec-POMDP). We introduced episodic memory into the reinforcement learning framework, proposing the CNN-Semantic Episodic Memory Utilization (CNN-SEMU) algorithm. Specifically, CNN-SEMU uses an encoder–decoder structure with a CNN to learn state embedding patterns influenced by the highest returns. It extracts semantic features from the high-dimensional map state space to construct a smoother memory embedding space, ultimately enhancing reinforcement learning performance by recalling the highest returns of historical states. Extensive simulation experiments demonstrate that in reconnaissance and search tasks of various scales, CNN-SEMU surpasses state-of-the-art multi-agent reinforcement learning methods in episodic rewards, search efficiency, and collision frequency.

Keywords: multi-UAV; limited communication range; uncertainty; reconnaissance and search; reinforcement learning; semantic episodic memory

1. Introduction

Advancements in flight control systems, endurance capabilities, and sensors have significantly expanded the use of Unmanned Aerial Vehicles (UAVs) in fields such as search and rescue (SAR) [1], military reconnaissance [2], and environmental monitoring [3]. Compared to traditional human-crewed aircraft or satellite reconnaissance methods, coordinating multiple UAVs (multi-UAV) for reconnaissance and search operations offers lower operational costs and greater flexibility. Additionally, UAVs can be equipped with various sensors to perform diverse reconnaissance and search tasks. Therefore, studying multi-UAV collaborative reconnaissance and search (MCRS) is highly important.

However, coordinating multi-UAVs for reconnaissance and search presents numerous challenges. First, effective communication between UAVs is necessary for information fusion. Thus, it is crucial to consider multi-UAV cooperative reconnaissance and search with a limited communication range (MCRS-LCR). Second, precise planning of each UAV’s flight path is required to ensure comprehensive search area coverage while avoiding collisions and path overlaps. Finally, UAVs need to make distributed decisions based on environmental changes and mission requirements, maintaining system robustness in case of a single-point failure. Additionally, target locations are often unknown and may be sparsely distributed over a vast, unexplored area [4]. If reconnaissance sensors have errors
and UAVs fail to detect targets on the first attempt, revisiting the area becomes challenging. These factors all contribute to the complexity of the MCRS-LCR problem.

To address the MCRS-LCR problem, current research primarily focuses on UAV cooperative control and search strategies. Cooperative control can be divided into centralized and distributed structures. Distributed control allows UAVs to make autonomous decisions without a command center, enhancing system scalability and resistance to interference. This has become the leading research direction in UAV control [5]. Regardless of the control structure, communication between UAVs is crucial for effective coordination. Oliehoek [6] categorizes communication among multiple agents into explicit, implicit, delayed, costly, and local communication. Based on these classifications, many researchers consider communication factors when exploring UAV target search problems [7–9]. However, for the MCRS-LCR problem, questions like “when to communicate” and “what to communicate” still require further investigation.

Regarding reconnaissance search strategies, some studies discretize the reconnaissance space and generate target probability maps to represent the probability distribution of target locations [7,8,10–12]. This method accounts for UAV reconnaissance sensor errors and performs well in practical applications. Other studies have developed optimization methods for multi-UAV systems based on heuristic algorithms [13], ant colony algorithms [14], and particle swarm algorithms [15]. However, as the number of UAVs increases and the environmental state space expands, these methods are prone to local optima.

Deep reinforcement learning (DRL) [16] offers powerful learning capabilities, providing new solutions to the MCRS-LCR problem. Multi-agent deep reinforcement learning (MADRL) further extends DRL’s application scope. MADRL commonly employs the Centralized Training with Decentralized Execution (CTDE) framework. During training, agents use global information and share knowledge to enhance collaboration. During execution, each agent chooses actions based on local observations, enabling distributed decision-making. This approach has succeeded significantly in StarCraft II [17], autonomous driving [18], and multi-robot formation [19].

In deep reinforcement learning, agents need to interact with the environment hundreds or thousands of times to reach human-level performance [20], resulting in data inefficiency. Unlike traditional reinforcement learning methods, human learning relies not only on interactions with the environment but also on past experiences stored in the hippocampus [21], a process known as episodic memory. However, recalling detailed episodes is often impossible. Instead, the human brain typically recalls features associated with high rewards. Episodic memory uses these features to guide future behavior (Figure 1). Research suggests that the ventral hippocampus may encode reward-related information and extract abstract features from environmental states [22–24], known as memory representation. This suggests that embedding semantic information with rewards into state space encoding can enhance the effectiveness of episodic memory.

Inspired by hippocampal episodic memory, Blundell et al. [25] proposed introducing episodic memory into reinforcement learning to improve learning efficiency. Zheng et al. [26], Hyungho et al. [27], and Ma [28] have used episodic memory in MADRL. However, the state space dimension in MADRL increases with the number of agents, making extracting semantic features from high-dimensional map state spaces in the MCRS-LCR problem particularly challenging. To efficiently utilize episodic memory by extracting semantic features from high-dimensional map state spaces, we draw an analogy to vision-based state spaces. An encoder–decoder structure with a CNN is used to learn state embedding patterns influenced by the highest returns. We call this method CNN-Semantic Episodic Memory Utilization (CNN-SEMU).

This paper explores the cooperative reconnaissance and search of unknown static targets using multiple UAVs with limited communication range. The main contributions are as follows:
1. A communication and information fusion model for the MCRS-LCR problem based on belief maps is proposed. Each UAV maintains a belief map for all UAVs and uses a max-plus-sum approach for information fusion, enabling effective communication.

2. Episodic memory is introduced into the MCRS-LCR problem. Using the value factorization centralized training with a decentralized execution (CTDE) framework, episodic memory leverages the highest state values stored in memory to generate better Temporal Difference targets, enhancing MADRL performance.

3. A new MADRL method called CNN-Semantic Episodic Memory Utilization (CNN-SEMU) is proposed. CNN-SEMU employs an encoder–decoder structure with a CNN to extract semantic features with the highest returns from high-dimensional map state spaces, enhancing the effectiveness of episodic memory.

![Figure 1. An example of episodic memory is a child’s encounter with an ice cream truck, a highly rewarding experience. When the child hears the truck’s music, they recall the previous scene of buying ice cream, including the music (a), the park location (b), the queue of customers (c), and the truck itself (d). The brain constructs episodic memories by recognizing these key features and associating them with positive experiences, guiding the child’s behavior, such as running outside when the music is heard.](image)

The rest of the paper is structured as follows: Section 2 introduces related work. Section 3 provides the system description and problem formulation. Section 4 reformulates the MCRS-LCR problem within the reinforcement learning framework. Section 5 introduces episodic memory into multi-UAV reinforcement learning and proposes the CNN-SEMU algorithm. Section 6 presents the experiments. Section 7 concludes the paper.

2. Related Work

This section briefly reviews current research on UAV communication modes, reconnaissance and search strategies, and episodic memory in reinforcement learning. It also identifies the gaps in these studies regarding the MCRS-LCR problem.

2.1. UAV Communication and Search Strategies

Communication between UAVs is crucial for efficient coordination. Explicit and implicit communication models do not consider communication costs, noise, and timing. In explicit communication [6], communication actions are added to the UAV’s action set, allowing UAVs to communicate by executing these actions. In contrast, implicit communication influences other UAVs’ observations through behavior without direct communication actions.
In [9], a communication topology matrix for UAVs is designed, represented as the Nth power of the UAV adjacency communication matrix. Communication between UAVs is achieved by exchanging digital pheromones. This method effectively represents the communication links between UAVs, but the matrix power calculation is computationally expensive. In [8], UAVs communicate by broadcasting the positive and negative detection times to their neighbors, taking the maximum value within the communication range. Although this method reduces communication volume, it does not accurately reflect the cumulative detection counts by neighboring UAVs. In [7], it is assumed that all UAVs are directly interconnected, simplifying calculations, but it may lead to network congestion with many UAVs communicating simultaneously. If UAV reconnaissance and search are likened to marking a chalkboard, implicit communication can be represented as other UAVs within the communication range instantly observing the mark. In this paper, we continue to study multi-UAV information fusion based on implicit communication.

The target probability map provides UAVs with probabilistic information about target distribution, which they use to optimize reconnaissance and search strategies. The studies [8,10,12] treat the existence of a target in each cell as a $0 - 1$ distributed random variable. The target probability map can be updated using Bayes' rule given an initial prior probability. However, if the prior probability is unavailable or incorrect, this method may affect the posterior probability calculation. The studies [7,11] use sensor readings as evidence sources and update the target probability map using Dempster–Shafer (DS) evidence theory. This method does not depend on prior probabilities and can effectively handle conflicting evidence when sensor readings are erroneous.

2.2. Episodic Memory in Reinforcement Learning

In single-agent reinforcement learning, Model-Free Episodic Control (MFEC) [25] uses episodic memory to solve sequential decision tasks. It stores the highest returns of explored state–action pairs in a tabular memory and greedily selects actions based on the maximum returns in the table. Since identical states are challenging to reproduce in natural environments, MFEC generalizes and embeds similar states using k-nearest neighbors and random projection techniques. Episodic Memory Deep Q-Networks (EMDQN) [29] use episodic memory to supervise agent training, combining the generalization strength of Deep Q-Networks (DQN) with the fast convergence of Episodic Control (EC), enabling quicker learning of better policies.

In multi-agent reinforcement learning, the state space dimension is usually more extensive, requiring adjustments and extensions to the feature embedding structures and learning frameworks from single-agent reinforcement learning. Zheng et al. [26] proposed Episodic Multi-agent Reinforcement Learning with Curiosity-driven Exploration (EMC), which introduces episodic memory into cooperative MADRL. EMC uses episodic memory to remember high-return states and employs one-step Temporal Difference (TD) memory targets for regularized learning. However, EMC still uses random projection for state embedding. According to the Johnson–Lindenstrauss lemma [30], random projection can approximately preserve the distance relationships of the original space. However, when the original state changes slightly, the embedded space may fluctuate significantly. Hyungho et al. [27] proposed Efficient Episodic Memory Utilization (EMU), which uses an encoder–decoder structure to learn semantic state embeddings, addressing the sparse selection of semantically similar memories caused by random projection. However, extracting semantic features directly from high-dimensional map state spaces is challenging. Additionally, EMU employs episodic incentives, providing extra rewards for states within desirable trajectories, thereby encouraging desirable state transitions. Desirable trajectories are defined as those that achieve the overall goal or exceed a preset reward threshold. However, the overall goal may be challenging in practical tasks, and the preset reward threshold requires additional domain knowledge. For the MCRS–LCR problem, if targets are sparsely distributed over a large area, it will be challenging to discover all targets, making additional episodic incentives less effective. In this paper, we continue to use regularized memory
targets to improve one-step TD learning and employ an encoder–decoder structure with a CNN to learn state embedding patterns influenced by the highest returns.

3. System Description and Problem Formulation

This section describes the system model, including the grid reconnaissance environment, the UAV model, the belief probability map model, and the UAV communication and information fusion model. Building on this, the MCRS-LCR is framed as a multi-objective optimization problem. Table 1 summarizes the main symbols used in this section.

Table 1. Main notations in Section 3.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{x,y} )</td>
<td>Cell at row ( x ) and column ( y ) in the map.</td>
</tr>
<tr>
<td>( u_{i,t} )</td>
<td>The position of UAV ( i ) at time step ( t ).</td>
</tr>
<tr>
<td>( z_k )</td>
<td>The position of the no-fly zone ( k ).</td>
</tr>
<tr>
<td>( N )</td>
<td>Number of UAVs.</td>
</tr>
<tr>
<td>( N_Z )</td>
<td>Number of no-fly zones.</td>
</tr>
<tr>
<td>( f_{x,y}(t) )</td>
<td>Sensor data from UAV ( i ) at time step ( t ) for ( c_{x,y} ).</td>
</tr>
<tr>
<td>( m_{t,x,y}(U), m_{t,x,y}(F), m_{t,x,y}(E) )</td>
<td>Basic probability assignments for the uncertainty and target maps at time step ( t ) for ( c_{x,y} ).</td>
</tr>
<tr>
<td>( m_b(U), m_b(F), m_b(E) )</td>
<td>Sensors’ basic probability assignments for the uncertainty and target maps.</td>
</tr>
<tr>
<td>( N_{t,x,y}^{(+)}, N_{t,x,y}^{(-)} )</td>
<td>At time step ( t ), the positive and negative detection times of UAV ( i ) for ( c_{x,y} ).</td>
</tr>
<tr>
<td>( m_f, m_e )</td>
<td>The belief that the sensor both identified and failed to identify the target.</td>
</tr>
<tr>
<td>( R_c )</td>
<td>The communication range of the UAV.</td>
</tr>
<tr>
<td>( Ne_{i,t} )</td>
<td>Neighbors of UAV ( i ) at time step ( t ).</td>
</tr>
<tr>
<td>( \tau )</td>
<td>The threshold at which the target is deemed to exist.</td>
</tr>
<tr>
<td>( d_1 )</td>
<td>No-fly zone safety distances.</td>
</tr>
<tr>
<td>( d_2 )</td>
<td>Safe distance for UAVs to avoid collisions.</td>
</tr>
</tbody>
</table>

3.1. Reconnaissance Environment Model

As shown in Figure 2, the task area is divided into \( L_1 \times L_2 \) square cells, with the cell in the \( i \)th row and \( j \)th column denoted as \( c_{x,y} \), where \( i \in \{1,2,\ldots,L_1\} \) and \( j \in \{1,2,\ldots,L_2\} \). It is assumed that several static targets are distributed within the task area, each occupying one cell, with at most one target per cell. Each cell has two possible states: empty \((E)\), indicating no target is present, or full \((F)\), indicating a target is present. Since there is no prior information about target locations, UAVs must use reconnaissance sensors to search for targets.

Additionally, specific cells in the map are assumed to be inaccessible to UAVs due to no-fly zones or obstacles. UAVs can avoid obstacles using a circular field of view (FOV). Let \( z_k \) represent the location of the \( k \)th no-fly zone cell, and assume no targets exist within these no-fly zones.

3.2. UAV Model

The distributed UAV system comprises \( N \) homogeneous UAVs that collaboratively perform reconnaissance and search tasks. The entire task process is divided into \( T \) time steps. At each time step, a UAV searches only the cell at its current position and decides the next step’s search direction, making the action space discrete. The size of each cell depends on the UAV’s reconnaissance performance and the time step setting. For example, if a UAV can complete the reconnaissance of a cell with 100-m sides in 10 min, then the cell size is set to 100 m, and the time step is set to 10 min.
Figure 2. System model: UAVs have eight possible movement directions. They coordinate reconnaissance and target search through a limited communication range with other UAVs while avoiding no-fly zones.

Each UAV has eight possible search directions at each time step. The position of a UAV can be identified by the cell it occupies. Let the position of UAV $i$ at time step $t$ be $u_{i,t}(x,y)$, which is updated by the chosen action $a$, represented as

$$
\begin{align*}
    u_{i,t+1}(x) &= u_{i,t}(x) + \Delta x \\
    u_{i,t+1}(y) &= u_{i,t}(y) + \Delta y,
\end{align*}
$$

where

$$
\Delta(x,y) = \begin{cases} 
    (-1, 0), & \text{if } a = 0 \text{ (left)} \\
    (1, 0), & \text{if } a = 1 \text{ (right)} \\
    (0, -1), & \text{if } a = 2 \text{ (down)} \\
    (0, 1), & \text{if } a = 3 \text{ (up)} \\
    (-1, -1), & \text{if } a = 4 \text{ (lower left)} \\
    (1, -1), & \text{if } a = 5 \text{ (lower right)} \\
    (1, 1), & \text{if } a = 6 \text{ (upper right)} \\
    (-1, 1), & \text{if } a = 7 \text{ (upper left)}.
\end{cases}
$$

If a UAV’s action would cause it to exceed the task area’s boundaries, that action is removed from the set of possible actions. This discrete action space modeling for UAVs eliminates the need to consider complex dynamics and control issues, such as motion...
posture and turning radius in continuous space, thereby simplifying the problem and improving path planning efficiency.

Assume all UAVs are equipped with identical reconnaissance sensors and each UAV completes the reconnaissance of its current cell within one step. If UAV $i$ performs reconnaissance in $c_{x,y}$ at time step $t$, the sensor reading is $f_{x,y}^i(t)$. If $f_{x,y}^i(t) = E$, a target is found in $c_{x,y}$; if $f_{x,y}^i(t) = F$, no target is found. Due to noise and other errors, the sensors may produce false readings, leading to conflicting reconnaissance results. To address this issue, we treat the sensor readings as evidence and use the Dempster–Shafer (DS) evidence theory, as established in [7], to handle these conflicts. By fusing the sensor readings, we can measure the uncertainty of each cell and represent the uncertainty of all cells as a belief probability map.

3.3. Belief Probability Maps

Given that each cell can be in one of two states, $E$ or $F$, the frame of discernment $\Lambda$ is defined as the power set of these states based on the DS evidence theory:

$$\Lambda = \{E, F, U\}$$

where $U = \{E, F\}$ represents the uncertain state. A probability can be assigned to each subset in the frame of discernment using the basic probability assignment (BPA) function $m$. This results in the belief probability map:

$$\sum_{A \in \Lambda} m_{x,y}(A) = m_{x,y}(E) + m_{x,y}(F) + m_{x,y}(U) = 1,$$

where the target map $m_{x,y}(F)$ and $m_{x,y}(E)$ represent the belief in the presence and absence of a target in $c_{x,y}$, respectively, while the uncertainty map $m_{x,y}(U)$ represents the uncertainty of $c_{x,y}$. Without prior information, the belief probability map is initialized as $m_{x,y}^{t=0}(E) = 0$, $m_{x,y}^{t=0}(F) = 0$, and $m_{x,y}^{t=0}(U) = 1$.

The core concepts of DS evidence theory are “evidence” and “combination”. “Evidence” refers to sensor readings containing uncertain information, while “combination” refers to the combination rules. According to DS combination rules, sensor readings can be fused as evidence into the belief map at the current time step, thereby updating the belief map for the next time step:

$$m_{x,y}^{t+1}(E/F/\Lambda) = m_{x,y}^t \oplus m_b(E/F/\Lambda)$$

$$= \frac{1}{k} \sum_{B \subseteq C \subseteq E/F/\Lambda} m_{x,y}^t(B) \times m_b(C), \forall B, C \in \Lambda,$$

where

$$k = 1 - \sum_{B \subseteq C \subseteq E/F/\Lambda} m_{x,y}^t(B) \times m_b(C), \forall B, C \in \Lambda,$$

and $m_b$ represents the sensor evidence’s BPA. When a sensor detects a target, this reading can be used as evidence to increase confidence in $F$. Since no information is provided about the absence of a target, the probability assigned to $E$ is 0. Due to sensor errors, this evidence is not entirely reliable, so the remaining probability is assigned to $U$. Thus, the evidence for detecting a target is represented as

$$\begin{cases} m_b(F) = m_f \\ m_b(E) = 0 \\ m_b(U) = 1 - m_f. \end{cases}$$
Likewise, the evidence for the sensor not detecting a target is represented as

\[
\begin{aligned}
    m_b(F) &= 0 \\
    m_b(E) &= m_e \\
    m_b(U) &= 1 - m_e,
\end{aligned}
\]  

(8)

where \( m_b \) and \( m_e \) represent the sensor’s confidence in detecting or not detecting a target, respectively. These values can be obtained from historical statistical data.

### 3.4. UAV Communication and Information Fusion Model

Each UAV maintains a distributed belief map and cannot access the global map. Therefore, UAVs must enhance their collaboration through information exchange and fusion. While \([7]\) assumes fully connected communication among UAVs, UAVs can only exchange information with their neighbors within the communication range. The neighbors of UAV \( i \) are defined as

\[
N_{e_{ij}} = \{ u_{j,i} | \| u_{j,i} - u_{i,i} \|_2 < R_c, j = 1, 2, \ldots, N, j \neq i \},
\]  

(9)

where \( R_c \) is the communication range of the UAV. The premise of information fusion is that the belief probability map is independent of the order of detection results. Therefore, we introduce positive and negative detection times. For each cell, a UAV records the positive detection times \( N_{i,y}^{df} (+) \) and the negative detection times \( N_{i,y}^{df} (-) \). Their update method is represented as

\[
\begin{aligned}
    N_{i,y}^{df+1} (+) &= N_{i,y}^{df} (+) + 1 \quad \text{if } f_{x,y}^i(t) = F \\
    N_{i,y}^{df+1} (-) &= N_{i,y}^{df} (-) + 1 \quad \text{if } f_{x,y}^i(t) = E.
\end{aligned}
\]  

(10)

Ref. \([7]\) demonstrated that the state of the belief map depends only on the positive and negative detection times, regardless of the order of detection results. Thus, the recursive formula in Equation (5) can be written to include only the positive and negative detection times. Each UAV can broadcast its positive and negative detection times to its neighbors within the communication range for information fusion, resulting in a distributed belief map for each UAV. It is important to note that our communication model assumes noiseless, instantaneous broadcast communication. This means each UAV broadcasts its messages and immediately receives messages from all other UAVs within its communication range without errors.

Assume each UAV maintains a map of the positive and negative detection times for all UAVs. When UAV \( j \) enters the communication range of UAV \( i \), UAV \( j \) merges its distributed map with UAV \( i \)'s distributed map. After information fusion, the positive and negative detection times for UAV \( i \) are represented as

\[
N_i = \sum_{q=1}^{N} \max(N_{i,q}, N_{j,q}), j \neq i.
\]  

(11)

In the above equation, we denote \( N_{i,q}^{df}(+/-) \) simply as \( N_{i,q} \) for conciseness. Equation (11) represents a process of taking the maximum value for each UAV and then summing them. For example, in a multi-UAV system with three UAVs, at time step \( t \), UAV \( A \)'s positive detection times for \( c_{x,y} \) are A3B1C0. This indicates that UAV \( A \)'s distributed map records \( c_{x,y} \) being detected by UAV \( A \) three times, by UAV \( B \) once, and by UAV \( C \) zero times. When UAV \( B \) enters \( A \)'s communication range with positive detection times \( A1B3C1 \) for \( c_{x,y} \), the information fusion updates UAV \( A \)'s positive detection times to \( A3B3C1 \), resulting in seven positive detections. In contrast, using the method from \([8]\), which takes the maximum detection times within the communication range, the fused positive detection count would be five, which does not accurately reflect the actual situation.
Denote \((1 - m_f)^{N_{ij}}(+)\) as \(f_x^j\) and \((1 - m_e)^{N_{ij}}(-)\) as \(e_x^j\). Under the initial conditions, the distributed target map and uncertainty map for UAV \(i\) after information fusion can be recursively calculated as

\[
\begin{align*}
\text{m}_{x,y}^j(E) &= \frac{f_x^j - e_x^j}{f_x^j + e_x^j} \\
\text{m}_{x,y}^j(F) &= \frac{e_x^j - e_x^j}{f_x^j + e_x^j} \\
\text{m}_{x,y}^j(U) &= \frac{1 + e_x^j}{f_x^j + e_x^j}.
\end{align*}
\]

### 3.5. Problem Formulation

The MCRS-LCR problem requires multi-UAV to reduce uncertainty in the task area and discover as many targets as possible with limited communication. UAVs must avoid no-fly zones and prevent collisions with other UAVs during flight. Therefore, a multi-objective optimization model is established as follows:

Objective function:

\[
\min \sum_{x=1}^{L_x} \sum_{y=1}^{L_y} m_{x,y}^{1:T}(U),
\]

\[
\max \sum_{x=1}^{L_x} \sum_{y=1}^{L_y} \mathbb{I}(m_{x,y}^{1:T}(F) \geq \tau),
\]

where \(\mathbb{I}\) is an indicator function that equals 1 when the condition inside the parentheses is met, and 0 otherwise. \(\tau\) represents the threshold for the presence of a target in a cell; when \(m_{x,y}^{1:T}(F) \geq \tau\), the cell is considered to contain a target. Equation (13) minimizes the uncertainty in the task area, while Equation (14) maximizes the number of targets discovered. These are complementary objective functions.

Constraints:

\[
\begin{align*}
0 &\leq u_{i,j}(x) \leq L_x \\
0 &\leq u_{i,j}(y) \leq L_y,
\end{align*}
\]

\[
\|z_k - u_{i,j}\|_2 > d_1, \forall k \in \{1, 2, \ldots, N_z\},
\]

\[
\|u_{i,j} - u_{i,j}\|_2 > d_2, i \neq j,
\]

where Equation (15) represents the UAV boundary constraint, while Equations (16) and (17) represent collision avoidance constraints. \(d_1\) and \(d_2\) are the safe distances for no-fly zones and UAV collision avoidance, respectively. Note that communication only affects the UAV’s observation space and is therefore not included in the constraints.

### 4. Reformulation

With a large number of UAVs and a vast state space dimension, conventional optimization methods can easily fall into local optima when solving the multi-objective optimization problem established in Section 3.5. Therefore, this section reformulates the multi-objective optimization problem within a decentralized partially observable Markov decision process (Dec-POMDP). Based on this, the state and action spaces of the UAVs are redefined, and a new reward function is designed according to the objectives and constraints of the original problem.

#### 4.1. Dec-POMDP

With limited communication, UAVs cannot observe the global state, necessitating the reformulation of the MCRS-LCR problem as a decentralized partially observable Markov decision process (Dec-POMDP) [6]. A Dec-POMDP is defined as a tuple \(M = (D, S, A, P, O, R, \gamma)\), where \(D = \{1, \ldots, N\}\) is the set of UAVs, \(S\) is the finite set
of environmental states, $A$ is the joint action set of multi-UAV, $P$ is the state transition function, $\Omega$ is the joint observation set, $O$ is the observation function, $R$ is the immediate reward, and $\gamma$ is the discount factor for rewards.

At each time step, each UAV $i$ makes a local observation $o_i \in \Omega$ and selects an action $a_i \in A$. The environment state $s \in S$ transitions to the next state $s'$ according to the state transition function $P(s'|s,a)$. The UAV team receives a shared immediate reward $r \in R(s,a,s')$. Under partially observable conditions, UAV $i$ can use its action-observation history $\tau_i \in \Gamma \equiv (\Omega \times A)$ to estimate the environment state. The joint action-observation history is denoted as $\tau \in \Gamma \equiv \Gamma^N$.

The policy of UAV $i$ is denoted as $\pi_i(a_i|\tau_i)$. The goal of cooperative MADRL is to learn a joint policy $\pi$ that maximizes the joint value function $V^\pi(\tau) = \mathbb{E}[\sum_{\tau_t=0}^\infty \gamma^t r_t \mid \tau_0 = \tau, \pi]$ or the joint action-value function $Q^\pi(\tau,a) = r + \gamma \mathbb{E}_{\tau'}[V^\pi(\tau')]$.

### 4.2. Observation, State, and Action Spaces

1. **Observation space:** The observation space refers to the portion of environmental information that a single UAV can directly perceive. For the MCRS-LCR problem, due to communication constraints, the observation space of UAV $i$ is represented as

   $$o_i = \left\{u_i(x)/L_x, u_i(y)/L_y, MapU_{\text{surround}}, MapF_{\text{surround}}, \text{RelPos}_{\text{agents}}, \text{RelPos}_{\text{traps}}\right\},$$

   where $u_i(x)/L_x$ and $u_i(y)/L_y$ represent the normalized coordinates of UAV $i$’s current position relative to the map’s maximum width. Normalizing to the range $[0,1]$ helps maintain scale consistency among different features.

   $MapU_{\text{surround}}$ and $MapF_{\text{surround}}$ are the distributed uncertainty map and target map $F$ of the eight surrounding movable cells of UAV $i$, respectively. Each is represented as an eight-dimensional vector:

   $$MapU_{\text{surround}} = \left\{m^I_{x+\Delta x,y+\Delta y}(U) \mid (\Delta x, \Delta y) \in D_s\right\},$$

   $$MapF_{\text{surround}} = \left\{m^I_{x+\Delta x,y+\Delta y}(F) \mid (\Delta x, \Delta y) \in D_s\right\},$$

   where $D_s$ defines the relative positions of the eight surrounding cells of UAV $i$:

   $$D_s = \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1)\}. \quad (21)$$

   The observation space includes only the belief maps of the eight surrounding movable cells of the UAV, not the entire local belief map. This reduces input dimensions and simplifies the UAV’s perception mechanism. If a direction is not movable, the corresponding observation vector is set to zero.

   $RelPos_{\text{agents}}$ and $RelPos_{\text{traps}}$ represent the relative positions of UAV $i$ to other UAVs and obstacles, respectively. Using relative positions helps improve the model’s ability to generalize in different environments. UAVs perceive other UAVs and obstacles through a circular field of view (FOV). To ensure fixed input dimensions, each UAV or obstacle is represented by a three-dimensional vector (distance, relative x-position, relative y-position). Vectors outside the FOV are filled with zeros.

2. **State space:** The state space provides a comprehensive view of the environment. For algorithms like QMIX [31] and QPLEX [32], the global state $s$ is used during centralized training to calculate the global action-value function. In the MCRS-LCR problem, the state space is represented as

   $$s = \left\{m(U), m(F), s_{\text{agents}}, s_{\text{traps}}, s_{\text{targets}}\right\}. \quad (22)$$
where \( m(U) \) and \( m(F) \) are the global belief maps, indicating the uncertainty and target presence on the map. \( s_{\text{agents}} \) represents the agent state, including the normalized positions of all UAVs. \( s_{\text{traps}} \) represents the no-fly zone state, including the normalized positions of all no-fly zones. \( s_{\text{targets}} \) represents the target state, including the normalized positions of all targets.

(3) Action space: The UAV’s action space includes eight possible actions. If an action would cause the UAV to exceed the task area boundaries, it is removed from the set of potential actions.

### 4.3. Reward Functions

In cooperative MADRL, a well-designed reward function can incentivize agents to collaborate and complete tasks. Based on the objective function and constraints of the MCRS-LCR problem, the reward function is designed as follows:

(1) Exploration reward: The exploration reward guides UAVs to explore the task area and reduce environmental uncertainty, thereby minimizing the uncertainty map. The exploration reward at time step \( t \) is given by

\[
r_{1,t} = \omega_1 \sum_{x=1}^{L_1} \sum_{y=1}^{L_2} (m^{t-1}_{x,y}(U) - m^{t}_{x,y}(U)).
\]

(2) Target discovery reward: For cells with contradictory reconnaissance results, UAVs must conduct multiple reconnaissances to strengthen the belief in the target presence. Therefore, the target discovery reward is given by

\[
r_{2,t} = \omega_2 \sum_{x=1}^{L_1} \sum_{y=1}^{L_2} I(m^{t}_{x,y}(F) \geq \tau) \times I((x,y) \notin D_{\text{find}}),
\]

where \( I \) is the indicator function, and \( D_{\text{find}} \) is the set of discovered targets. The reward is given only when a target is found for the first time.

(3) Collision prevention: A penalty is given when a UAV enters the no-fly zone:

\[
r_{3,t} = -\omega_3 \sum_{i=1}^{N} \sum_{k=1}^{N_z} I(\|u_{i,t} - z_k\|_2 \leq d_1).
\]

To avoid collisions between UAVs, a penalty is given when the distance between them is less than the safe distance:

\[
r_{4,t} = -\omega_4 \sum_{i=1}^{N} \sum_{j=i+1}^{N} I(\|u_{i,t} - u_{j,t}\|_2 \leq d_2).
\]

Therefore, the total reward at time step \( t \) can be expressed as

\[
r_t = r_{1,t} + r_{2,t} + r_{3,t} + r_{4,t}.
\]

### 5. Methodology

This section introduces the CNN-Semantic Episodic Memory Utilization (CNN-SEMU) algorithm (Figure 3), which comprises six modules. First, we explain how to construct and update the episodic buffer. Next, we highlight the shortcomings of current embedding methods in addressing the MCRS-LCR problem. Then, we incorporate the CNN module into the state embedding network by drawing an analogy to vision-based state spaces. Finally, we detail the algorithm’s training process.
5.1. Value Factorization CTDE Framework

Centralized training with decentralized execution (CTDE) is a commonly used learning paradigm in multi-agent systems [33, 34]. This approach also applies to the MCRS-LCR problem reformulated under the Dec-POMDP framework. During the centralized training phase, all UAVs collaborate using global information to learn and determine the optimal joint action-value function $Q_{\text{tot}}^*(\tau, a; \theta)$. The parameter $\theta$ is learned by minimizing the one-step TD loss:

$$L(\theta) = E_{\tau, a, \tau', s} \left[ (y(\tau, a) - Q_{\text{tot}}(\tau, a; \theta))^2 \right],$$

where $y(\tau, a) = r + \gamma \max_{a'} Q_{\text{tot}}(\tau', a'; \theta^-)$ is the one-step TD target, $D$ is the replay buffer, and $\theta^-$ is the target network parameter.

During the decentralized execution phase, due to partial observability of the environment, each UAV can only access its local observation history and make decisions based on its action-value function $Q_i(\tau_i, a_i)$. To achieve this, value decomposition is typically used. According to the IGM principle [35], the joint action-value function $Q_{\text{tot}}$ is decomposed into the individual contributions $Q_i(\tau_i, a_i)$ of each agent.

5.2. Episodic Memory

Figure 3 illustrates the framework of the CNN-Semantic Episodic Memory Utilization (CNN-SEMU) algorithm. Figure 3a shows the standard value factorization CTDE module, where the Mixing Network combines each agent’s action-value function into the joint action-value function $Q_{\text{tot}}$ using a function $f$. The function $f$ can be implemented using methods like QMIX [31], QPLEX [32], and VDN [36]. Figure 3b depicts the replay buffer, which stores the trajectories $(r, \tau, a, \tau', s)$ generated from agent–environment interactions. In addition to the replay buffer, CNN-SEMU includes an episodic buffer (Figure 3c). The episodic buffer stores the highest returns $H(s)$ of historical states, used for the regularization part of the one-step TD loss in the value factorization module, expressed as

$$L_{\text{memory}}(\theta) = E_{\tau, a, \tau', s} \left[ (H(s) - Q_{\text{tot}}(\tau, a; \theta))^2 \right].$$
However, as the number of agents increases, the state space of MADRL grows exponentially, requiring a vast amount of memory to directly use the high-dimensional state space. Therefore, researchers typically do not use the global state $s$ vector space, which aligns with how memory is represented in the human brain [24]. The use of computer vision techniques, such as convolutional neural network (CNN), to extract semantics in such a high-dimensional state space is very challenging.

When the replay buffer collects a new trajectory with timestamp $t$, the $H(x_t)$ in the episodic buffer is updated:

$$H(x_t) = \begin{cases} \max\{H(\hat{x}_t), R_t(s_t, a_t)\}, & \text{if } \|\hat{x}_t - x_t\|_2 < \delta \\ R_t(s_t, a_t), & \text{otherwise,} \end{cases}$$

(30)

where $\hat{x}_t$ is the nearest neighbor to $x_t$ found in the episodic buffer using the nearest neighbor algorithm [37], and $\delta$ is the threshold for measuring the approximate relationship between $\hat{x}_t$ and $x_t$. $R_t(s_t, a_t)$ represents the return given the global state $s_t$ and joint action $a_t$.

The construction and update rules for the episodic buffer are given in Algorithm 1. First, a trajectory is collected from the replay buffer (line 1). Then, a reverse chronological traversal is used to calculate the return $R_t$ (lines 2–4). Next, the encoder computes $x_t$ (line 5), and the nearest neighbor algorithm finds the closest $\hat{x}_t$ in the current episodic buffer (line 6). Last, lines 7–13 handle the update and addition of memories. If the embedded state $x_t$ is sufficiently close to an existing embedded state $\hat{x}_t$ (within the threshold $\delta$) and $x_t$ has a higher return $R_t$, the reward in memory is updated (line 9), and the embedded state in memory is replaced with the new embedded state (line 10: memory shift). If the embedded state $x_t$ is outside the threshold $\delta$, it indicates that $x_t$ is a new state not present in memory and is added to the episodic buffer (line 13).

**Algorithm 1 Construction and Update of Episodic Buffer $D_E$**

1. $T = \{r_0, \cdots, r_T, s_0, \cdots, s_T\}$: trajectory
2. Initialize $R_{T+1} = 0$.
3. for $t = T$ to 0 do
4. set $R_t = r_t + \gamma \times R_{t+1}$.
5. Compute $x_t = \phi(s_t)$.
6. Use the nearest neighbor algorithm to select the closest neighbor, $\hat{x}_t \in D_E$.
7. if $\|\hat{x}_t - x_t\|_2 < \delta$ then
8. if $H(\hat{x}_t) < R_t$ then
9. $H(\hat{x}_t) \leftarrow R_t$
10. $\hat{x}_t \leftarrow x_t$, $\hat{s}_t \leftarrow s_t$ (memory shift)
11. end if
12. end if
13. Add memory $D_E \leftarrow (x_t, H(x_t), s_t)$
14. end for

For the state embedding function $\phi(s)$, EMC [26] uses random projection. While this method effectively reduces dimensionality, it imposes random weights on the state vectors, causing them to be randomly distributed in the embedding space. This reduces the range of the threshold $\delta$, potentially preventing the recall of similar states with higher returns and only recalling identical states. EMU [27] proposes a trainable encoder–decoder structure to extract important features with reward semantics from the global state. Although the embedding space is smoother compared to random projection, the state space dimension of MCRS-LCR is enormous (as indicated by Equation (22)). Extracting features with return semantics in such a high-dimensional state space is very challenging.

In vision-based environments such as autonomous driving [38] and robotic navigation [39], the input to the state space is typically images or video frames. Agents can use computer vision techniques, such as convolutional neural network (CNN), to extract
image features and understand the current environmental state. Both images and grid maps discretize the environment into small units (pixels or grids), each with specific values representing certain environmental features. Inspired by this, convolution operations can extract features from grid maps. Additionally, Equation (4) indicates that the belief map comprises three parts: \( E, F, \) and \( U \), whose sum always equals 1. If each map is considered a channel, there must be correlations between these channels.

Figure 3d shows the state embedding network structure of CNN-SEMU. As illustrated, we split the state into the map state \( s_{map} \) and the other parts of the state \( s_{other} \). The \( s_{map} \) includes the target map \( F \) and the uncertainty map \( U \), which are processed through a CNN to extract grid map features. The \( s_{other} \) is processed directly through a fully connected network (FC), and the extracted features are combined with the grid map features extracted by the CNN to form the final embedded state \( x \). The decoder reconstructs the embedded state \( x \) into a mirror image \( \hat{s} \) of the original state \( s \) and outputs a predicted reward \( \mathcal{H} \) through another head to ensure that the embedded state \( x \) contains information about the highest return. The final loss function is expressed as

\[
\mathcal{L}(\phi, \psi) = \mathbb{E}_{s, H \in D_E}[\text{max}(H - \psi_{H}(\phi(s | t)))^2 + \lambda_1 ||s - \psi_t(\phi(s | t)||^2_2)], \tag{31}
\]

where \( \phi(s | t) \) is the encoder state embedding function, with the additional input of time step \( t \) to improve feature extraction quality [27]. \( D_E \) is the episodic buffer, \( \psi_{H} \) represents the decoder branch predicting the highest return, \( \psi_t \) represents the decoder branch reconstructing the global state \( s \), and \( \lambda_1 \) is the scaling factor that adjusts the ratio between the prediction loss and reconstruction loss.

When the encoder–decoder structure is updated, the mapping relationship between the original and embedded states \((\phi(s) : S \rightarrow R^k)\) and the key–value pair mapping \((R^k \rightarrow H)\) change. Therefore, it is necessary to store the state key–value pairs \((s, H(x), x)\) in the episodic buffer and periodically update the embeddings in the episodic buffer (update embedding).

Figure 3f shows the CNN network structure used to extract map features. Using a 20 \times 20 \times 2 map as an example, the first and second convolutional layers use 3 \times 3 kernels with a stride of 2 to extract primary and higher-level features from the input map. The third layer uses a 5 \times 5 kernel with a stride of 1 to cover a larger receptive field, integrating the local features extracted by the previous layers into a more global feature representation.

Figure 3e visualizes the utilization of episodic memory, where the rainbow-colored circles from red to purple represent embedded states with low to high return values. During model training, a batch of global states \( s_t \) with timestamps is sampled from the replay buffer and encoded into \( x_t \) through state embedding. Then, \( x_t \) queries past experiences in the episodic buffer. Using an encoder–decoder structure with CNN allows the embedded space to maintain a more approximate positional structure to the original state space. Therefore, \( x_t \) can retrieve memories from \( \tilde{s}_t \) within a larger threshold \( \delta \), efficiently utilizing episodic memory.

The total loss function for training the \( Q_{tot} \) network is expressed as

\[
\mathcal{L}_\theta = \mathbb{E}_{\tau, a, s \in D}[(y(\tau, a) - Q_{tot}(\tau, a; \theta))^2 + \lambda (H(\phi(s), a) - Q_{tot}(\tau, a; \theta))^2], \tag{32}
\]

where \( H(\phi(s), a) = r_t + \gamma H(\phi(s_{t+1})) \) is the one-step TD memory target and \( \lambda \) is the regularization factor.

5.3. Model Training

Algorithm 2 outlines the training process of CNN-SEMU. In line 7, the value factorization CTDE algorithm includes methods such as VDN, QMIX, and QPLEX. In line 8, \( t_{emb} \) represents the update period for the episodic buffer. Line 11, update embedding, indicates that when the encoder–decoder structure is updated, all embedded states \( x \) in the episodic
buffer need to be updated immediately to maintain the matching relationship between key–value pairs.

**Algorithm 2 CNN-SEMU**

1. Initialize parameters $\theta$, $\theta^-$, $\phi$ and $\psi$.  
2. **while** episode $\leq \text{max-episode}$$\text{do}$  
3. UAVs interact with the environment using $\varepsilon$–greedy algorithm based on $Q^\theta_s$, execute an episode, and obtain a trajectory $T$.  
4. Run Algorithm 1 to update the episodic buffer $D_E$.  
5. Store $T$ in replay buffer $D_E$.  
6. Sample a random batch of trajectories $T_{i=1}^{M_1}$ from $D$ with batch size $M_1$.  
7. Run the value factorization CTDE algorithm with $T_{i=1}^{M_1}$ and update $\theta$ using Equation (32).  
8. **if** $t_{env}$ mod $t_{emb}$ == 0 **then**  
9. Sample a random batch of $(s, H, t)_{i=1}^{M_2}$ from $D_E$ with batch size $M_2$.  
10. update $\phi$ and $\psi$ using Equation (31).  
11. update all $x \in D_E$ with new $\phi$ (Update embedding)  
12. **end if**  
13. **end while**

**6. Experiments**

This section evaluates the effectiveness of the model and algorithm through extensive experiments. First, the experimental setup is introduced. Next, the impact of communication conditions on reconnaissance search effectiveness is analyzed. Then, CNN-SEMU is compared with other baselines, and the effect of state embedding on algorithm performance is examined. Finally, key parameters influencing the algorithm are analyzed through parameter experiments, and their impact on the embedding space is discussed.

**6.1. Experimental Environment Setup**

To ensure the generality of the results, we evaluated the algorithm on maps of three different sizes: $10 \times 10$, $15 \times 15$, and $20 \times 20$, with detailed explanations provided for the $20 \times 20$ map. The specific environmental settings are shown in Table 2. To ensure target discovery, the episodic length was set so that the UAVs could cover the task area at least twice.

**Table 2.** Map scene settings.

<table>
<thead>
<tr>
<th>Map Size</th>
<th>Number of UAVs</th>
<th>Number of Targets</th>
<th>Number of No-Fly Zones</th>
<th>Dimension of State Space</th>
<th>Dimension of Observation Space</th>
<th>Dimension of Action Space</th>
<th>Episodic Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10 \times 10$</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>226</td>
<td>36</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>$15 \times 15$</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>502</td>
<td>57</td>
<td>8</td>
<td>70</td>
</tr>
<tr>
<td>$20 \times 20$</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>880</td>
<td>75</td>
<td>8</td>
<td>100</td>
</tr>
</tbody>
</table>

In each episode, the positions of the UAVs, targets, and no-fly zones are initialized randomly. During UAV movement, the positions of the targets and no-fly zones are unknown and fixed. UAVs can detect the positions of no-fly zones and other UAVs through their FOV. Other parameters used in the experiments are shown in Table 3. To better compare with baseline algorithms, the episodic latent dimension follows the settings in [26,27], and the reconstruction loss scaling factor $\lambda_1$ follows the settings in [27].
Table 3. Experimental parameter settings.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAV communication range ($R_c$)</td>
<td>5</td>
</tr>
<tr>
<td>FOV radius ($R_{\text{FOV}}$)</td>
<td>4</td>
</tr>
<tr>
<td>No-fly zone safety distances ($d_1$)</td>
<td>1</td>
</tr>
<tr>
<td>Collision safety distance between UAVs ($d_2$)</td>
<td>1</td>
</tr>
<tr>
<td>$m_f$</td>
<td>0.8</td>
</tr>
<tr>
<td>$m_e$</td>
<td>0.8</td>
</tr>
<tr>
<td>$\omega_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_2$</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_3$</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_4$</td>
<td>1</td>
</tr>
<tr>
<td>Episodic latent dimension ($\text{dim}(x)$)</td>
<td>4</td>
</tr>
<tr>
<td>Episodic buffer capacity</td>
<td>1 M</td>
</tr>
<tr>
<td>Target discovery threshold ($\tau$)</td>
<td>0.95</td>
</tr>
<tr>
<td>Reconstruction loss scale factor ($\lambda_1$)</td>
<td>0.1</td>
</tr>
<tr>
<td>Regularized scale factor ($\lambda$)</td>
<td>0.1</td>
</tr>
<tr>
<td>State-embedding difference threshold ($\delta$)</td>
<td>0.0013</td>
</tr>
</tbody>
</table>

The experiments were conducted on a server with multiple NVIDIA GeForce RTX 4090 GPUs, each with 24 GB of memory and 90 GB of RAM. Unless otherwise specified, each experiment data point averages 160 samples, with 32 episodes tested per random seed during training, using five random seeds.

6.2. Communication Model Analysis

To test the effectiveness of the proposed UAV communication and information fusion model, this section analyzes the performance of multi-UAV cooperative reconnaissance under different communication conditions. The communication model was tested using the standard QMIX [31] in a 20 × 20 map.

Figure 4 shows the distributed uncertainty map $U$ of UAV $i$ under different communication conditions, depicted using contour maps. Redder colors indicate higher uncertainty, while bluer colors indicate lower uncertainty. Compared to heat maps, contour maps more clearly describe the numerical changes in uncertainty across the map. Figure 4a,b shows the uncertainty maps with communication, where UAV $i$ achieves compelling environmental exploration through real-time information sharing. Over time, the overall map becomes bluer, indicating reduced uncertainty. In contrast, Figure 4c,d shows the uncertainty maps without communication, where UAV $i$ is unaware of the reconnaissance results of other UAVs, leaving most of the map red. The scattered blue areas on the right side of the map indicate no-fly zones. Comparing Figure 4a and Figure 4c at the same time shows that UAV $i$ in Figure 4c is likely to continue exploring the red areas already covered by other UAVs, while UAV $i$ in Figure 4a uses the communication mechanism to gain a comprehensive understanding of the entire map, effectively avoiding overlapping trajectories and improving reconnaissance efficiency.

Figure 5 tests the impact of communication distance on the number of targets discovered and the level of uncertainty. As the communication distance increases, the number of targets discovered gradually rises, and uncertainty gradually decreases, eventually converging to a stable value. This indicates that increasing the communication distance enhances information exchange opportunities among UAVs, improving their cooperative reconnaissance and search capabilities. In the 20 × 20 map, when the communication distance of the UAVs reaches 5, the number of targets discovered and the level of uncertainty have already converged to optimal values, indicating that a larger communication distance is not necessary to achieve the best communication effect.
6.3. Baseline Experiment Analysis

CNN-SEMU can be integrated with any value decomposition-based MADRL framework. In this paper, we implemented CNN-SEMU based on the QMIX algorithm. The experiments compared EMC and EMU, both using QMIX as the basic framework, and included state-of-the-art MADRL algorithms like QMIX and QPLEX as baselines. Additionally, we conducted an ablation study by removing the CNN structure from CNN-SEMU (denoted as SEMU).
Figure 6 compares the episodic rewards of CNN-SEMU and baseline algorithms on 10 × 10 and 15 × 15 maps. The results show that CNN-SEMU achieves the best performance in smaller task scenarios.

![Figure 6](image1)

(a) 4UAV, 10 × 10 map.  
(b) 8UAV, 15 × 15 map.

**Figure 6.** Comparison of CNN-SEMU and baseline algorithms on episodic reward.

Figure 7 shows the experimental results of 10 UAVs on 20 × 20 map. The experiment compared four metrics: episodic reward, uncertainty, number of collisions, and number of targets found. As shown in Figure 7a, CNN-SEMU achieved the best performance, and all three episodic memory-based methods outperformed QMIX and QPLEX, demonstrating the effectiveness of episodic memory. QPLEX had the fastest convergence in uncertainty and the number of targets found in the early stages of training but the slowest convergence in the number of collisions. Additionally, CNN-SEMU converged the fastest and to better values in uncertainty, the number of collisions, and the number of targets found.

![Figure 7](image2)

(a) Episodic reward.  
(b) Uncertainty.  
(c) Number of collisions.  
(d) Number of targets found.

**Figure 7.** Performance comparison of CNN-SEMU and baseline algorithms on 20 × 20 Map.
Episodic methods improve sampling efficiency by recalling the highest returns of historical states and using one-step memory TD loss to estimate the action–state value function more quickly and accurately. This results in better performance compared to non-episodic QMIX and QPLEX. As shown in Figure 7a, SEMU, which only uses the encoder–decoder structure, performs worse than EMC with random projection in the later stages of training. This is because it is challenging to directly extract features in the high-dimensional (880) map state space. In contrast, CNN-SEMU uses CNN to extract map features, achieving semantically meaningful embeddings that can recall similar states with higher returns. It is worth noting that CNN-SEMU initially performs worse than QMIX in the early stages of training, as the embedding network has not yet captured the important features of the high-dimensional map state space.

Comparison with EMU

EMU also uses an encoder–decoder structure for state embedding. However, unlike the one-step TD memory target, EMU introduces episodic incentives to provide additional rewards for states on desirable trajectories. Episodic incentives are defined as

\[ r_p = \gamma \frac{N_{q}(s')}{N_{call}(s')} \left( H(f_p(s')) - \max_{a'} Q_{\theta}(s', a') \right), \]

where \( N_{call}(s') \) represents the total number of times state \( s' \) is visited in the episodic buffer, and \( N_{q}(s') \) represents the number of desirable visits. The final loss function is expressed as

\[ L(\theta) = \mathbb{E}_{\tau,a,r \in D} \left[ (r_p + y(\tau, a) - Q_{tot}(\tau, a; \theta))^2 \right]. \]

EMU defines desirable trajectories as those with episodic rewards exceeding a preset threshold \( R_{thr} \). In StarCraft II, the maximum reward is obtained when all enemy units are eliminated, and [27] sets this maximum reward as the threshold \( R_{thr} \). For the MCRS-LCR problem, discovering all targets is challenging, making it difficult to determine the maximum overall episodic reward. In the comparative experiments, using a 20 × 20 map as an example, we set \( R_{thr} \) to 365 and 370 based on the results in Figure 7a. Additionally, we set a periodically updated \( R_{thr} \) that can be adjusted based on the episodic rewards tested during training. This allows states initially deemed desirable to be re-evaluated as undesirable as cognition updates. However, when the desirability of states in the episodic buffer changes, it becomes difficult to determine \( N_{q}(s') \). Therefore, we omit \( N_{q}(s') / N_{call}(s') \) from Equation (33) and denote this method as EMU-change.

In the comparative experiments with EMU, we used the same CNN-based embedding structure. As shown in Figure 8a, CNN-SEMU exhibited the best performance, while EMU–365/370 performed similarly to QMIX. This is because EMU only receives episodic incentives when the recalled state is desirable, resulting in EMU–365/370 rarely receiving the \( r_p \) signal during training. Figure 8b shows that EMU–365/370 maintains a similar average \( Q - value \) to QMIX, indicating that the episodic incentive \( r_p \) did not take effect. Figure 8b displays the \( Q_{tot} \) average value for all UAVs. It is evident that the EMU-change has a much higher average \( Q - value \) than the other algorithms, indicating it received the \( r_p \) signal. However, as shown in Figure 8a, the EMU-change performs worse than QMIX. This may be because, although the desirability update enhances the episodic incentive effect, it overestimates the action–state value function. In contrast, CNN-SEMU can quickly and accurately estimate the action-state value function.
6.4. State Embedding Analysis

This subsection examines how the proposed algorithm alters the embedding space, affecting performance. Figure 9 shows the t-SNE [40] visualization results of 20 K randomly sampled state samples from a 1M memory episodic buffer in a 20 × 20 map task. Since the state vectors have no labels, we label each state vector with its corresponding highest return $H$.

Figure 9a shows the state vectors before embedding. State vectors with similar colors are clustered together, indicating that state vectors with similar returns are closely arranged in space before embedding. Figure 9b shows the embedded state space using EMC random
projection. Although random projection approximately preserves the distance relationships between state vectors, the embedded states are scattered in space with almost no clustering effect. Figure 9c shows the embedding representation of CNN-SEMU without the CNN structure, achieving primary clustering but still less effective than in Figure 9d.

CNN-SEMU achieves good clustering of similar return embeddings, resulting in a smoother embedding space. This indicates that the encoder–decoder with a CNN structure can effectively extract features with reward semantics. Additionally, this clustering makes selecting episodic memory around $x_t$ safer (Figure 3e), ensuring that the memories queried within a certain threshold are closely related states. This allows a larger $\delta$ to be chosen, effectively utilizing more memories.

6.5. Hyperparametric Analysis

Selecting appropriate hyperparameters is essential for model training. This section examines the impact of critical hyperparameters $\lambda$ and $\delta$ on algorithm performance. The hyperparameter experiments are conducted using a $20 \times 20$ map as an example.

(1) Sensitivity analysis of regularization factor $\lambda$

$\lambda$ controls the proportion of one-step TD memory loss in Equation (32). In this subsection, we examine the stability of CNN-SEMU’s performance under different settings of $\lambda$ ($[0.01, 0.05, 0.1, 0.5, 1]$). As shown in Figure 10, when $\lambda$ is 0.01, 0.05, or 0.1, CNN-SEMU achieves optimal performance. However, when $\lambda$ is too large (0.5 or 1), the algorithm’s performance degrades significantly. This may be because excessive recall of historical states causes the algorithm to get stuck in local optima.

![Figure 10. The effect of hyperparameter $\lambda$ on CNN-SEMU performance.](image)

(2) Sensitivity analysis of the state-embedding difference threshold $\delta$

$\delta$ influences the update and usage of the episodic buffer during training through Equation (30). In this section, we study the impact of different $\delta$ values on three episodic-based methods: EMC, SEMU, and CNN-SEMU. EMC constructs the embedding space using random projection, CNN-SEMU uses an encoder–decoder structure with CNN, and SEMU is a variant of CNN-SEMU without the CNN structure.

Ref. [27] proposed a method to determine the appropriate threshold $\delta$, expressed as

$$\delta \leq \frac{2 \times 3\sigma_x}{\text{dim}(x)} \cdot \frac{\text{M}}{M},$$

(35)

where $M$ is the capacity of the episodic buffer, and $\sigma_x$ represents the standard deviation of $x$ in the episodic buffer. When $\text{dim}(x) = 4$, $M = 1 \times 10^6$, and $\sigma_x$ is approximately...
1, $\delta$ can be calculated as $\delta \leq 0.0013$. Therefore, in this subsection, we set the $\delta$ range to $[1.3 \times 10^{-4}, 1.3 \times 10^{-3}, 1.3 \times 10^{-2}, 1.3 \times 10^{-1}]$ to study its impact on episodic-based methods.

Figure 11 shows the episodic reward at the end of training for different $\delta$ settings, with the horizontal axis on a logarithmic scale. From Figure 11, it is evident that CNN-SEMU achieves the best training performance across various $\delta$ values and performs well over a broader range of $\delta$. When $\delta = 1.3 \times 10^{-1}$, the episodic reward for EMC drops sharply. This is because EMC uses random projection to construct the embedding space, resulting in a disordered distribution (as shown in Figure 9b). Within the $\delta = 1.3 \times 10^{-1}$ range, it is possible to retrieve states with significantly different reward values, and this excessive recall can disrupt the training process.

![Figure 11. Episodic reward at the end of training for different values of $\delta$.](image)

To demonstrate both training efficiency and effectiveness, Table 4 lists the episodic reward based on training time, and Figure 12 shows the training curves for different $\delta$ values. As shown in Table 4, CNN-SEMU, which uses an encoder–decoder structure with CNN, converges faster and to a higher episodic reward in most cases. This indicates that in the smoother state space constructed by CNN-SEMU, it is safe to retrieve memories from a wider range of $\delta$. However, when $\delta = 1.3 \times 10^{-1}$, CNN-SEMU’s performance declines, suggesting that although CNN-SEMU is robust to a wide range of $\delta$, excessively large $\delta$ can still recall completely unrelated states. Therefore, selecting an appropriate $\delta$ helps improve algorithm performance. When $\delta = 1.3 \times 10^{-3}$, CNN-SEMU converges to the highest value, indicating that the choice of $\delta$ in Equation (35) is reasonable.

### Table 4. The effect of $\delta$ on episodic reward according to training time.

<table>
<thead>
<tr>
<th>Timesteps (M)</th>
<th>0.67</th>
<th>1.33</th>
<th>2.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta$</td>
<td>EMC</td>
<td>SEMU</td>
<td>CNN-SEMU</td>
</tr>
<tr>
<td>$1.3 \times 10^{-4}$</td>
<td>355.69</td>
<td>363.22</td>
<td>359.17</td>
</tr>
<tr>
<td>$1.3 \times 10^{-3}$</td>
<td>356.03</td>
<td>358.62</td>
<td>361.91</td>
</tr>
<tr>
<td>$1.3 \times 10^{-2}$</td>
<td>359.43</td>
<td>358.84</td>
<td>359.77</td>
</tr>
<tr>
<td>$1.3 \times 10^{-1}$</td>
<td>339.95</td>
<td>343.10</td>
<td>337.03</td>
</tr>
</tbody>
</table>

The optimal results of each parameter experiment are highlighted in bold.

As shown in Figure 12, a larger $\delta$ results in higher episodic reward variance than a smaller $\delta$. When $\delta = 1.3 \times 10^{-1}$, both EMC and SEMU fail to find the optimal strategy and exhibit larger error bands, validating our previous analysis. Although larger $\delta$ can
recall more similar states, it may also recall completely unrelated states. In contrast, CNN-SEMU uses semantic embedding with CNN, clustering states with the same reward in the embedding space. This makes the space smoother and reduces the likelihood of recalling unrelated states.

Figure 12. The effect of $\delta$ on episodic reward.

7. Conclusions and Future Work

This paper focuses on the multi-UAV cooperative reconnaissance and search problem with a limited communication range (MCRS-LCR) for unknown static targets. We propose a cooperative communication and information fusion model based on belief maps. For the MCRS-LCR problem, we introduce a new multi-agent reinforcement learning method based on episodic memory called CNN-SEMU. CNN-SEMU uses an encoder–decoder structure with CNN to construct the embedding space, extracting high-return semantic features from the high-dimensional map state space, making the embedding space smoother. Ultimately, CNN-SEMU improves reinforcement learning sampling efficiency through one-step memory TD targets, thereby enhancing learning. Extensive baseline simulations demonstrate that CNN-SEMU outperforms state-of-the-art multi-agent reinforcement learning methods in episodic rewards, search efficiency, and collision frequency. State embedding and parameter sensitivity analyses reveal the reasons for CNN-SEMU’s superior performance. This is due to the encoder–decoder with a CNN structure, which achieves better semantic state embeddings.

Future research will optimize communication methods between UAVs to reduce unnecessary load. We will also consider constraints such as communication delay and bandwidth to address practical challenges. Currently, our research focuses on the reconnaissance and search of static targets. Moving forward, we will extend our research to include dynamic target reconnaissance and address the challenges of transitioning from simulation to reality.
Author Contributions: Conceptualization, B.Z. and T.W.; methodology, B.Z., M.L. and Y.C.; software, B.Z.; data curation, B.Z.; writing—original draft preparation, B.Z.; writing—review and editing, X.L. and Z.Z.; visualization, B.Z. and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the National Natural Science Foundation of China (62003359) and the Basic Strengthening Plan Project of China (2023-JCJQ-JJ-0795).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References
5. Xia, J.; Zhou, Z. The Modeling and Control of a Distributed-Vector-Propulsion UAV with Aero-Propulsion Coupling Effect. Aerospace 2024, 11, 284. [CrossRef]
6. Oliehoek, F.A.; Amato, C. A Concise Introduction to Decentralized POMDPs; SpringerBriefs in Intelligent Systems, Springer International Publishing: Cham, Switzerland, 2016. [CrossRef]
19. Li, J.; Liu, Q.; Chi, G. Distributed deep reinforcement learning based on bi-objective framework for multi-robot formation. Neural Netw. 2024, 171, 61–72. [CrossRef] [PubMed]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.