Deep Reinforcement Learning for Multi-UAV Exploration Under Energy Constraints

Yating Zhou[1], Dianxi Shi[2(✉)], Huanhuan Yang[1], Haomeng Hu[1], Shaowu Yang[1], and Yongjun Zhang[2]

[1] College of Computer, National University of Defense Technology, Changsha 410073, Hunan, China
[2] Artificial Intelligence Research Center (AIRC), National Innovation Institute of Defense Technology (NIIDT), Beijing 100071, China
dxshi@nudt.edu.cn

Abstract. Autonomous exploration is the essential task for various applications of unmanned aerial vehicles (UAVs), but there is currently a lack of available energy-constrained multi-UAV exploration methods. In this paper, we propose the RTN-Explorer, an environment exploration strategy that satisfies the energy constraints. The goal of environment exploration is to expand the scope of exploration as much as possible, while the goal of energy constraints is to make the UAV return to the landing zone before the energy is exhausted, so they are a pair of contradictory goals. To better balance these two goals, we use map centering, and local-global map processing methods to improve the system performance and use the minimum distance penalty function to make the multi-UAV system satisfy the energy constraints. We also use the map generator to generate different environment maps to improve generalization performance. A large number of simulation experiments verify the effectiveness and robustness of our method and show superior performance in benchmark comparison.

Keywords: Multi-UAV exploration · Deep reinforcement learning · Energy constraints

1 Introduction

Autonomous exploration means that the agent, without any prior knowledge, keeps moving in a new environment and constructs a map of the whole environment. It is an essential part of many tasks, such as planetary exploration [1], reconnaissance [2], rescue [3], mowing [4], and cleaning [5]. Compared with single-agent environment exploration, multi-agent environment exploration is more

This work was partially supported by the National Natural Science Foundation of China No. 91948303.

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 461, pp. 363–379, 2022.
https://doi.org/10.1007/978-3-031-24386-8_20

difficult because of cooperation between agents. Despite decades of research, multi-agent environment exploration, as an NP-hard problem [6], remains a complex problem to solve.

Many traditional methods have been proposed in recent years, but these methods rely heavily on experts to manually design heuristic functions for different scenes [7,8]. Research on deep reinforcement learning methods gradually increased [9–16]. However, the current environment exploration strategies based on deep reinforcement learning only focus on exploration efficiency and do not consider energy constraints. Due to the limited battery capacity of UAVs, it is necessary to consider energy constraints for tasks in the real world.

Considering the energy constraints in a real exploration task, the UAV needs to return to the landing zone before the energy is exhausted, which will bring great challenges to the multi-agent exploration task. The first challenge is to design an efficient reward function to ensure that the multi-agent system satisfies the energy constraints. In [17], the energy constraint is considered in the path planning task of a given environment, and a constant penalty is given to the agent when the UAV runs out of energy. But the constant value penalty function is difficult to make the UAV incline to return to the landing area. The second challenge is that meeting the energy constraints and improving the exploration rate are in conflict. Achieving a balance between the two goals requires well-designed solutions. The third challenge is the need to stabilize the performance of multi-UAV exploration systems on different maps.

Based on the above facts, this paper proposes RTN-Explorer, a multi-UAV exploration strategy under energy constraints, while exploring the unknown environment as much as possible while meeting energy constraints. We introduce the minimum distance penalty function. The UAV obtains a penalty proportional to the shortest distance to the landing zone to ensure that the UAV meets the energy constraint. We design a DDQN-based network architecture and introduce map centering, global-local map processing to improve the performance of the multi-UAV exploration system. We also implemented a map generator, which can generate different environment maps for training to improve generalization. We have carried out many experiments and verification to prove the effectiveness of our proposed method. Our contributions are summarized as follows:

1) We introduce energy constraints into exploration tasks based on deep reinforcement learning for the first time. Our minimum distance penalty function effectively improves the return rate to more than 93%, which is 92% higher than the return rate of the constant penalty.
2) We design a DDQN-based network architecture and introduce map centering, global-local map processing to improve the performance of the multi-UAV exploration system. We increase the exploration rate to more than 92.85%.
3) To improve the generalized performance of the multi-UAV system, we design a map generator that can randomly change the position and size of obstacles.

2 Related Work

Autonomous robotic exploration has always been a hot research topic in robotics because of its applications in rescue tasks. Depending on whether deep reinforcement learning techniques are used, existing exploration methods are categorized as 1) classical or 2) deep reinforcement learning.

2.1 Classical Methods for Environment Exploration

Classical methods utilize handcrafted heuristics to allocate goal locations to robots to maximize exploration efficiency [6]. To date, the most common method used for exploring unknown environments is frontier exploration. Zhou et al. [18] introduce a frontier information structure to generate efficient global coverage paths, which completes the exploration tasks with unprecedented efficiency (3–8 times faster) compared to other single robot approaches. But multi-robot exploration is a more effective way to improve the efficiency of exploration. In the multi-robot frontier exploration method [19], each agent makes its own decision to select a target to explore based on the shared frontier. And the multi-robot frontier exploration was improved by ranking the agents to allocate to a particular frontier location based on their distances to the frontier [20]. And Lopez-Perez et al. utilize a distributed multi-robot model to increase robustness [21]. In addition, some works [22–24] improve the practicality of multi-robot exploration by considering inter-robot communication and cooperation.

2.2 DRL Methods for Environment Exploration

Deep reinforcement learning(DRL) methods can enable agents to learn complex exploration strategies through repeated interactions with the environment, thereby improving their decision-making abilities [9]. Many existing DRL approaches only focus on single UAV scenarios. Niroui et al. [10] proposed to combine deep reinforcement learning with frontier exploration. They use deep reinforcement learning to learn exploration strategies and then use traditional navigation methods to complete exploration tasks. Koutras et al. [11] provided a framework for learning exploration/coverage policies that possess strong generalization abilities due to the procedurally generated terrain diversity. However, in the above work, the autonomous exploration strategy is only suitable for single-agent scenarios, which severely limits the efficiency and robustness of the exploration system.

Using multiple agents has several advantages, such as reducing task completion time, improving the fault tolerance of the whole system, and so on. This motivates the need for further research on multi-agent collaborative exploration. In [13], a hierarchical control architecture for networked explorers is proposed. A Voronoi-based exploration algorithm and deep reinforcement learning-based collision avoidance approach are then provided to coordinate the robots efficiently while avoiding sudden obstacles. He et al. [14] proposed a distributed multi-robot exploration algorithm based on deep reinforcement learning (DME-DRL) for structured environments that enables robots to make decisions based on this

high-level knowledge. DME-DRL is a distributed algorithm that uses deep neural networks to extract the structural pattern of the environment, and it can work in scenarios with or without communication. Geng et al. [15] presented the attention-based communication neural network (CommAttn) to "learn" the cooperation strategies automatically in the decentralized multi-robot exploration problem. The communication neural network enables the robots to learn cooperation strategies with explicit communication. Moreover, the attention mechanism can precisely calculate whether the communication is necessary for each pair of agents by considering the relevance of each received message, which enables the robots to communicate only with the necessary partners.

2.3 Summary of Limitations

In summary, classical methods utilize hand-crafted heuristics that require domain expert knowledge and extensive manual tuning of utility and cost parameters to achieve expected cooperative behavior [25]. But DRL methods no longer require hand-crafted features/functions. It can learn cooperative policies directly from agent experience. However, existing DRL methods do not take energy constraints into account.

As far as the author knows, the current exploration tasks are mainly used in scenarios such as reconnaissance [2] and rescue [26]. In these scenarios, disregarding the energy constraints is impossible for the UAV because the energy that the UAV can carry is limited. This requires the UAV to return and land before the energy is exhausted. To address these limitations, we introduce RTN-Explorer, the first multi-UAV DRL method that considers energy constraints and UAV return, which uniquely designs input processing, network structure, and energy-constrained rewards. And training in different environments ensures generalization.

3 Problem Formulation

We define the multi-UAV exploration problem as a team of UAVs $I = \{i_1, \dots i_n\}$ cooperating to explore an unknown environment and generate a global map $\mathcal{G}$. To simplify the problem, we represent the environment as a grid graph $\mathcal{M}$ containing $M \times M$ cells of size c. The set $\mathcal{L}$ represents the start/landing positions, and $\mathcal{L}$ is given by Eq. (1). The lowercase letters l, b, g correspond to their respective environment representation $\mathcal{L}$, $\mathcal{B}$, $\mathcal{G}$.

$$\mathcal{L} = \left\{ \left[x_i^l, y_i^l\right]^{\mathrm{T}}, i = 1, \dots, L, \quad : \left[x_i^l, y_i^l\right]^{\mathrm{T}} \in \mathcal{M} \right\} \tag{1}$$

And the set $\mathcal{B}$ of the positions of the obstacles that the UAVs cannot occupy is given by Eq. (2).

$$\mathcal{B} = \left\{ \left[x_i^b, y_i^b\right]^{\mathrm{T}}, i = 1, \dots, B, \quad : \left[x_i^b, y_i^b\right]^{\mathrm{T}} \in \mathcal{M} \right\} \tag{2}$$

The global map $\mathcal{G}$ composed of the exploration area is given by Eq. (3).

$$\mathcal{G} = \left\{ \left[x_i^g, y_i^g\right]^{\mathrm{T}}, i = 1, \dots, G, \quad : \left[x_i^g, y_i^g\right]^{\mathrm{T}} \in \mathcal{M} \right\} \tag{3}$$

3.1 UAV Model

For any UAV $i \in I$, the position of the i-th UAV at time t is defined as $\mathbf{p}_i(t) = [x_i(t), y_i(t), z_i(t)]^{\mathrm{T}} \in \mathbb{R}^3$ with $z_i(t) \in \{0, h\}$. It means that the UAV is either at ground level or in constant altitude h. In addition to the position of the UAV, the state of the i-th UAV includes operating status $\phi_i(t) \in \{0, 1\}$ and battery energy $b_i(t) \in \mathbb{N}$. The action space of each UAV can be defined as Eq. (4).

$$\mathcal{A} = \{ \underbrace{\begin{bmatrix}0\\0\\0\end{bmatrix}}_{\text{hover}}, \underbrace{\begin{bmatrix}c\\0\\0\end{bmatrix}}_{\text{east}}, \underbrace{\begin{bmatrix}0\\c\\0\end{bmatrix}}_{\text{north}}, \underbrace{\begin{bmatrix}-c\\0\\0\end{bmatrix}}_{\text{west}}, \underbrace{\begin{bmatrix}0\\-c\\0\end{bmatrix}}_{\text{south}}, \underbrace{\begin{bmatrix}0\\0\\-h\end{bmatrix}}_{\text{land}} \} \tag{4}$$

The action of the i-th UAV at time t is $\mathbf{a}_i(t) \in \tilde{\mathcal{A}}(\mathbf{p}_i(t))$. $\tilde{\mathcal{A}}(\mathbf{p}_i(t))$ is defined by Eq. (5). The UAV can only perform the landing action in the landing zone, otherwise it can only perform the other five actions.

$$\tilde{\mathcal{A}}(\mathbf{p}_i(t)) = \begin{cases} \mathcal{A}, & \mathbf{p}_i(t) \in \mathcal{L} \\ \mathcal{A} \backslash [0, 0, -h]^{\mathrm{T}}, & \text{otherwise} \end{cases} \tag{5}$$

Assume that the UAV can only move one unit distance c in each time slot δ_t. The speed of each UAV is $v_i(t) \in \{0, V\}$, which means that the UAV is either moving at speed $V = c/\delta_t$ or stationary. The position transformation method after the action is executed is shown in Eq. (6). The UAV's position can only be changed when its operating status is active ($\phi_i(t) = 1$).

$$\mathbf{p}_i(t+1) = \begin{cases} \mathbf{p}_i(t) + \mathbf{a}_i(t), & \phi_i(t) = 1 \\ \mathbf{p}_i(t), & \text{otherwise} \end{cases} \tag{6}$$

The transition function of the UAV's operational status is given by Eq. (7). If the UAV is inactive at time t or performs the landing action, it is inactive at time $t+1$.

$$\phi_i(t+1) = \begin{cases} 0, & \mathbf{a}_i(t) = [0, 0, -h]^{\mathrm{T}} \vee \phi_i(t) = 0 \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

The change of the UAV's remaining energy is represented by Eq. (8). If the state of the i-th UAV at time t is active, then the energy at time $t+1$ is reduced by one. Otherwise, it remains unchanged.

$$b_i(t+1) = \begin{cases} b_i(t) - 1, & \phi_i(t) = 1 \\ b_i(t), & \text{otherwise} \end{cases} \tag{8}$$

The size of the combined area that all the UAVs can explore at time t is given by Eq. (9), where $D_i(t)$ represents the size of the i-th UAV's exploration area at time t.

$$G(t) = \sum_{i=1}^{I} D_i(t+1) - \sum_{i=1}^{I} D_i(t) \tag{9}$$

3.2 Optimization Problem

The objective of the above problem is to maximize the joint exploration area while satisfying the energy constraint. The maximization problem is given by Eq. (10) optimizing over joint actions $\times_i \mathbf{a}_i(t)$. At the same time, the following five constraints must be satisfied. The first constraint is that each active UAV cannot be in the same position as the other active UAVs to avoid collisions. The second constraint is that the UAV cannot collide with obstacles. The third constraint is that the UAV's residual energy is always greater than or equal to 0. The last two constraints ensure that the UAV is initially in the start/landing zone, active, and at height h.

$$\begin{aligned} &\max_{\times_i \mathbf{a}_i(t)} \sum_{t=0}^{T} G(t) \\ &\text{s.t. } \mathbf{p}_i(t) \neq \mathbf{p}_j(t) \vee \phi_j(t) = 0, && \forall i, j \in I, i \neq j, \forall t \\ &\quad \mathbf{p}_i(t) \notin \mathcal{B}, && \forall i \in \mathcal{I}, \forall t \\ &\quad b_i(t) \geq 0, && \forall i \in \mathcal{I}, \forall t \\ &\quad \mathbf{p}_i(0) \in \mathcal{L} \wedge z_i(0) = h, && \forall i \in \mathcal{I} \\ &\quad \phi_i(0) = 1, && \forall i \in I \end{aligned} \tag{10}$$

3.3 UAV System

Figure 1 is a system-level diagram depicting the sensors and software components of a UAV. The UAV is equipped with a localization module and a scanning camera for exploring the environment. The map processing module generates the current map and feeds it to the reinforcement learning agent. Each UAV is initialized with a fixed movement budget which is its initial energy. The safe controller is responsible for translating the RL agent's proposed action into a safe action, and we will introduce the conversion method in Sect. 4.

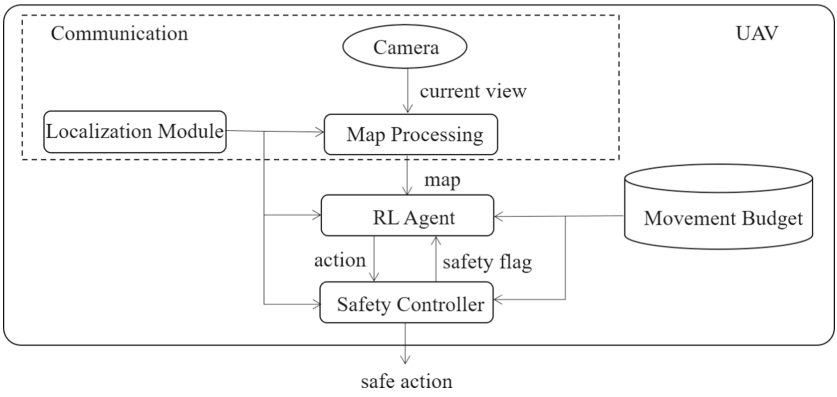


Fig. 1. System-level diagram depicting sensor and software components on the UAV during an multi-UAV exploration task.

4 Deep Reinforcement Learning Framework

In this section, we convert the multi-robot exploration problem under energy constraints into a decentralized partially observable Markov decision process (Dec-POMDP) [27]. The network architecture is shown in Fig. 2. We break down the observations into a map of the start/landing zone, a map of the exploration area, UAV's positions and battery information, and a map of obstacles in the current view. The decomposed views are then subjected to map processing, including map centering and global-local map processing. The global and local maps are fed through convolutional layers with ReLU activation and fully-connected networks to extract features. Then we calculate the next action and reward through the DDQN model trained in Sect. 4.3.

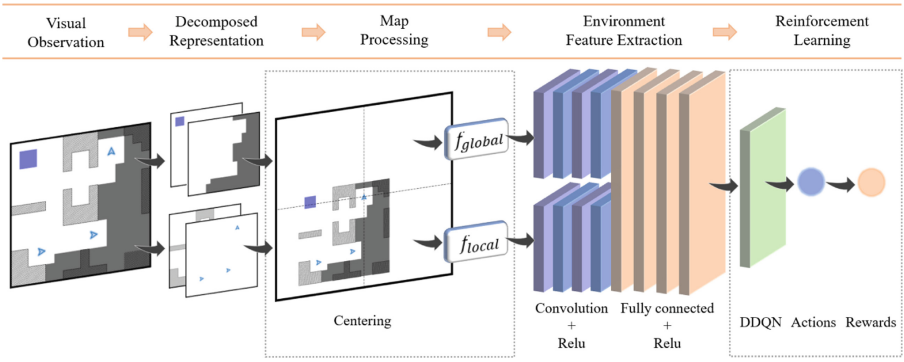


Fig. 2. The architecture of the RTN-Explorer. In decomposed representation, the information of obstacles and the explored area is incrementally increased during the unknown environment exploration.

4.1 Markov Decision Process

The Dec-POMDP is defined through the tuple $(\mathcal{S}, \mathcal{A}_\times, T, R, \Omega_\times, \mathcal{O}, \gamma)$. In the Dec-POMDP, $\mathcal{S}$ represents the state space, $\mathcal{A}_\times$ represents the joint action space and T is the transition probability function. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ represents the reward function that maps the current state, action and next state to a real number representing the reward. $\Omega_\times = \Omega^I$ is the joint observation space. $\mathcal{O} : \mathcal{S} \times \mathcal{I} \mapsto \Omega$ represents the observation function that map state and agents to one agent's individual observation. γ is a discount factor, which represents the trade-off between current and future returns.

State Space. The state space of the multi-UAV exploration problem under energy constraints is given by Eq. (11), where the state $s(t) \in \mathcal{S}$ at time t is given by Eq. (12). $\forall i \in \mathcal{I}$, $\mathbf{M} \in \mathbb{B}^{M \times M \times 3}$ is the tensor representation of the set of start/landing zones $\mathcal{L}$, explored area $\mathcal{G}$, and obstacles $\mathcal{B}$. The other elements

of the tuple $\mathcal{S}$ represent positions, remaining flying times, and operational status of all agents.

$$\mathcal{S} = \underbrace{\mathcal{L}}_{\substack{\text{Landing}\\\text{Zones}}} \times \underbrace{\mathcal{G}}_{\substack{\text{Explored}\\\text{Area}}} \times \underbrace{\mathcal{B}}_{\text{Obstacles}} \times \underbrace{\mathbb{R}^{I\times 3}}_{\substack{\text{UAV}\\\text{Positions}}} \times \underbrace{\mathbb{N}^{I}}_{\substack{\text{Flying}\\\text{Times}}} \times \underbrace{\mathbb{B}^{I}}_{\substack{\text{Operational}\\\text{Status}}} \tag{11}$$

$$s(t) = \left(\mathbf{M}, \{\mathbf{p}_i(t)\}, \{b_i(t)\}, \{\phi_i(t)\}\right) \tag{12}$$

Safety Controller. Figure 1 shows the architecture of the UAV, which includes the safety controller. As shown in Eq. (13), the safety controller rejects an action with a safety threat and converts the action into a hover action while preserving the safe action. Safety-threatening maneuvers include collisions with other UAVs, collisions with obstacles, and landings in non-landing zones.

$$\mathbf{a}_{s,i}(t) = \begin{cases} [0,0,0]^{\mathrm{T}}, & \mathbf{p}_i(t) + \mathbf{a}_i(t) \in \mathcal{B} \\ & \vee\, \mathbf{p}_i(t) + \mathbf{a}_i(t) = \mathbf{p}_j(t) \wedge \phi_j(t) = 1, \quad \forall j, j \neq i \\ & \vee\, \mathbf{a}_i(t) = [0,0,-h]^{\mathrm{T}} \wedge \mathbf{p}_i(t) \notin \mathcal{L} \\ \mathbf{a}_i(t), & \text{otherwise.} \end{cases} \tag{13}$$

Reward Function. The reward of the i-th UAV at time t is calculated by Eq. (14). $D_i(t+1) - D_i(t)$ represents the difference between the exploration range of two adjacent moments, and α is the weight parameter of the reward obtained by the exploration. ϵ represents the punishment caused by energy consumption.

$$r_i(t) = \alpha \left(D_i(t+1) - D_i(t)\right) + \beta_i(t) + \gamma_i(t) + \epsilon \tag{14}$$

$\beta_i(t)$ is the penalty value when the RL agent's proposed action is rejected by the security controller. β is a hyperparameter.

$$\beta_i(t) = \begin{cases} \beta, & \mathbf{a}_i(t) \neq \mathbf{a}_{i,s}(t) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

$\gamma_i(t)$ is the penalty for the UAV not landing in the landing zone when the remaining energy is zero. Unlike the method calculated in [17], we do not use a constant as a penalty here but a value proportional to the minimum distance between the UAV and the landing zone. In Eq. (16), $\mathbf{s}$ represents the position vector of the center of the landing zone and λ is a multiplier times the minimum distance. Relative to the constant value penalty, using $\gamma_i(t)$ as the penalty can impose different penalties for landing in a non-landing zone depending on the distance so that the UAV gets less penalty when it is closer to the landing zone.

$$\gamma_i(t) = \begin{cases} \lambda \times dis(\mathbf{p}_i(t), \mathbf{s}), & b_i(t+1) = 0 \wedge \mathbf{p}_i(t+1) = [\cdot, \cdot, h]^{\mathrm{T}} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

4.2 Map Processing

The map processing methods we introduced include map centering and global-local map processing. Map centering is the transformation of the input map of each UAV into an expanded map with the UAV as the center. It can make the network pay more attention to the information closely related to the current UAV and avoid the influence of invalid details. Localized processing is to crop the centered map and retain the middle part. The localization makes the network pay more attention to the information near the UAV to assist the action decision. Globalized processing is to compress the centered map and extract its features.

Map Centering. For input map $\mathbf{A} \in \mathbb{R}^{M\times M\times n}$, we utilize Eq. (17) for centering to get $\mathbf{B} \in \mathbb{R}^{M_c\times M_c\times n}$, where $M_c = 2M-1$. $\tilde{\mathbf{p}}$ is the position vector of the current UAV and $\mathbf{x}_{\text{pad}}$ is the fill value of the augmented map.

$$\mathbf{B} = f_{\text{center}}\ (\mathbf{A}, \tilde{\mathbf{p}}, \mathbf{x}_{\text{pad}}\ ) \tag{17}$$

The function f_{center} is given by Eq. (18).

$$f_{\text{center}}\ : \mathbb{R}^{M\times M\times n} \times \mathbb{N}^2 \times \mathbb{R}^n \mapsto \mathbb{R}^{M_c\times M_c\times n} \tag{18}$$

The calculation formula for each element in $\mathbf{B}$ is shown in Eq. (19). It effectively pads map $\mathbf{A}$ with the padding value $\mathbf{x}_{\text{pad}}$.

$$\mathbf{b}_{i,j} = \begin{cases} \mathbf{a}_{i+\tilde{p}_0-M+1,j+\tilde{p}_1-M+1}, & M \le i+\tilde{p}_0+1 < 2M \\ & \wedge M \le j+\tilde{p}_1+1 < 2M \\ \mathbf{x}_{\text{pad}}\ , & \text{otherwise} \end{cases} \tag{19}$$

Localized Processing. Localized processing transforms $\mathbf{B} \in \mathbb{R}^{M_c\times M_c\times n}$ into $\mathbf{X}$ according to the parameter l.

$$\mathbf{X} = f_{\text{local}}\ (\mathbf{B}, l) \tag{20}$$

The function f_{local} is given by Eq. (21).

$$f_{\text{local}}\ : \mathbb{R}^{M_c\times M_c\times n} \times \mathbb{N} \mapsto \mathbb{R}^{l\times l\times n} \tag{21}$$

Each element in $\mathbf{X}$ is calculated as shown in Eq. (22). $\mathbf{X}$ is obtained by intercepting the middle $l \times l$ part on $\mathbf{B}$.

$$\mathbf{x}_{i,j} = \mathbf{b}_{i+M-\lceil \frac{l}{2} \rceil, j+M-\lceil \frac{l}{2} \rceil} \tag{22}$$

Globalized Processing. Globalized processing transforms $\mathbf{B} \in \mathbb{R}^{M_c\times M_c\times n}$ into $\mathbf{Y}$ according to the parameter g.

$$\mathbf{Y} = f_{\text{global}}\ (\mathbf{B}, g) \tag{23}$$

The function f_{glocal} is given by Eq. (24).

$$f_{\text{global}} : \mathbb{R}^{M_c \times M_c \times n} \times \mathbb{N} \mapsto \mathbb{R}^{\left\lfloor \frac{M_C}{g} \right\rfloor \times \left\lfloor \frac{M_C}{g} \right\rfloor \times n} \tag{24}$$

The elements in $\mathbf{Y}$ are defined by Eq. (25). This operation equals an average pooling operation with pooling cell size g.

$$\mathbf{y}_{i,j} = \frac{1}{g^2} \sum_{u=0}^{g-1} \sum_{v=0}^{g-1} \mathbf{b}_{gi+u,gj+v} \tag{25}$$

4.3 Multi-agent Reinforcement Learning

Deep Q-network (DQN) is a Q-learning algorithm based on deep learning, which mainly combines value function approximation and neural network [28]. It adopts the method of target network and experience replay to train the network. Experience replay builds a replay buffer $\mathcal{D}$. New experiences of the agent, represented by quadruples of (s, a, r, s'), are stored in the replay buffer. r represents the reward obtained by performing action a after state s and s' represents the next state. DQN uses a separate target network to estimate the next largest Q value. In order to solve the problem of DQN overestimating the Q value, DDQN improves the target value as:

$$Y^{\text{DDQN}}(s, a, s') = r(s, a) + \gamma Q_{\bar{\theta}} \left(s', \underset{a'}{\text{argmax}} Q_\theta(s', a') \right) \tag{26}$$

And its loss function is given by:

$$L^{\text{DDQN}}(\theta) = \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[\left(Q_\theta(s, a) - Y^{\text{DDQN}}(s, a, s') \right)^2 \right] \tag{27}$$

During training, the sampled soft-max policy for exploration of the state and action space is given by Eq. (28). The hyperparameter β is used to balance exploration and exploitation.

$$\pi(a_i \mid s) = \frac{e^{Q_\theta(s,a_i)/\beta}}{\sum_{\forall a_j \in \mathcal{A}} e^{Q_\theta(s,a_j)/\beta}} \tag{28}$$

The DDQN training process is described in Algorithm 1. Following the initialization of the replay buffer and network parameters, new training begins to reset the state, select a random UAV starting position, and a random mobile budget $b_0 \in \mathcal{B}$ for each UAV. As long as the exploration task is not completed, the event will continue. For each activate UAV i, a new action $a \in \mathcal{A}$ is chosen according to Eq. (28) and the subsequent experience stored in the replay memory buffer $\mathcal{D}$. The main network parameter θ is updated by utilizing the ADAM optimizer to execute gradient steps on data with a small batch of m samples in the replay buffer. Subsequently, updating target network parameter $\bar{\theta}$ and reducing the mobile budget. The exploration task ends when all the UAVs have successfully landed or are at zero power. Then, a new episode begins, unless the maximum number of episodes is $N_{\max}$.

Algorithm 1. DDQN training for exploreation under energy constraints

input: maximum training steps N_{max}, initial movement budget range $\mathcal{B}$
output: network parameter θ

Initialize D, initialize θ randomly, $\bar{\theta} \leftarrow \theta$
for $t = 0$ to N_{max} **do**
  $\forall i \in \mathcal{I}$, initialize the UAV's state s_i with random starting position and sample initial movement budget b_i uniformly from $\mathcal{B}$
  while the task is not completed **do**
    for each UAV $i \in \mathcal{I}$ **do**
      if the i-th UAV is inactivate **then** continue
      Sample a_i according to Eq. (28)
      Observe r_i, s_i'
      Store (s_i, a_i, r_i, s_i') in $\mathcal{D}$
      for $j = 1$ to m **do**
        Sample $\left(s_j, a_j, r_j, s_j'\right)$ uniformly from $\mathcal{D}$
        $Y_j = \begin{cases} r_j, & \text{if } s_j' \text{ terminal} \\ \text{according to } Eq.\,(27), & \text{otherwise} \end{cases}$
        Compute loss $L_j(\theta)$ according to Eq. (26)
      Update θ with gradient loss $\frac{1}{m}\sum_{j=1}^{m} L_j(\theta)$
      $\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$
      $b_i = b_i - 1$

5 Experiments

In this section, we evaluate different exploration strategies under different conditions. We conduct simulation experiments on a variety of maps generated by the map generator. The exploration rate and the return rate are the metrics we use to evaluate the agents' performance on different maps and under different scenario instances. We define the exploration rate as the ratio between the amount of explored area and the total area of the ground truth map. The return rate equals the number of UAV successful returns divided by the total number of trials.

5.1 Experiment Setup

The algorithm of DDQN is implemented with TensorFlow [29] and the group of UAVs with a scanning camera. We train the multi-UAV coordinated exploration on a computer with an Intel Xeon W-2235 CPU and an NVIDIA GeForce RTX 3090 GPU. The environment is presented as 32×32 cells space. As shown in Fig. 2, each unit in the environment is assigned to one object: obstacle (grey), the unexplored region (dark grey), the start/landing zone (purple), and UAV (blue). In the training experiment, we assume each UAV can only move one cell within one cell scanning range at a time in the environment. We reinitialize the exploration scene when the UAVs are reaching the maximal steps or completing all the tasks. To demonstrate the robustness of our model, we randomly place the

UAVs in the start/landing zones, generating random environments for the UAVs at each re-initialization. We set the number of UAVs to be a random number in the range of $[2, 6]$, and the energy of the UAVs to be a random number in the range of $[400, 450]$. The key hyperparameters during the training process are listed in Table 1.

Table 1. DDQN hyperparameters

Parameter	Value	Description
$\|\theta\|$	1,175,302	trainable parameters
$N_{\max}$	3,000,000	maximum training steps
l	17	local map scaling
g	3	global map scaling
$\|\mathcal{D}\|$	50,000	replay buffer size
m	128	minibatch size
$\boldsymbol{\tau}$	0.005	soft update factor
γ	0.95	discount factor in Eq. (26)
β	0.1	temperature parameter in Eq. (28)
λ	0.2	the minimum distance multiplier in Eq. (16)

5.2 Experiment Results

Comparison of Different Map Processing Methods. To prove the validity of our proposed map processing method, we compared four different map processing methods. Table 2 shows the performances of the exploration systems using four different map processing methods when the number of UAVs ranges from [2, 6]. As for the evaluation, Table 2 shows the results from the experiments as averaged over 5000 runs for each different method. We also use different maps generated by the map generator during the test to ensure the reliability of the results.

Increasing the exploration rate and increasing the return rate are conflicting goals. In our experiment, the weight of the exploration reward is greater than the weight of the return failing penalty, so the UAVs will tend to explore the unknown area. This is why no processing method makes the exploration system obtain more than 86% exploration rate and less than 1% return rate. Only using global-local map processing can improve the return rate to a certain extent but reduce a certain exploration rate. Only using map centering can significantly improve the return rate. Combining map centering and global-local map processing, we can get the highest return rate and exploration rate. This is because the local map processing intercepts the central part of the UAV view, strengthens the information around the UAV, and globalizes the centralized view to extract the global features. As the number of UAVs increases, so does the scale of the problem, which leads to a certain reduction in the return rate of UAVs.

Table 2. Performance comparison of different map processing methods.

Map Processing\UAVs		2	3	4	5	6
No processing	Exploration rate	86.25%	90.91%	94.77%	95.18%	96.54%
	Return Rate	0.00%	0.10%	0.22%	0.72%	0.47%
Global-Local	Exploration Rate	83.59%	90.68%	94.22%	95.53%	96.17%
	Return Rate	2.28%	3.66%	3.77%	3.39%	2.75%
Centering	Exploration Rate	87.82%	93.25%	93.68%	93.23%	92.44%
	Return Rate	13.52%	33.55%	64.10%	72.84%	73.87%
Global-Local +Centering	Exploration Rate	**92.85%**	**96.00%**	**97.67%**	**98.40%**	**98.84%**
	Return Rate	**93.10%**	**98.23%**	**97.30%**	**94.16%**	**93.01%**

Comparison of Different Return Strategies. To verify that our return strategy can explore new areas while returning, we compare the exploration rate with the return strategy based on the original path. The strategy of return based on the original path is that when the energy consumption of the UAV reaches half of the initial energy, the UAV returns to the landing zone.

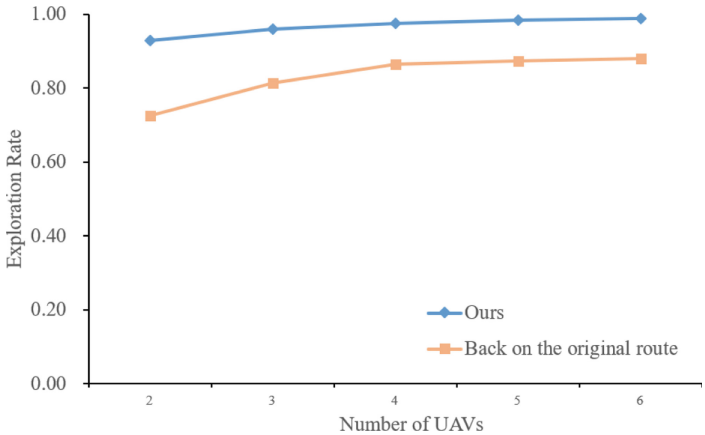


Fig. 3. Comparison of exploration efficiency between RTN-Explorer and the return strategy based on the original path.

As shown in Fig. 3, the abscissa represents the number of robots. We compared the exploration rates of two return strategies with the number of UAVs in [2,6]. It can be seen from Table 1 that our exploration strategy can ensure a high return rate of more than 93%. In terms of exploration rate, our strategy is at least 10.88% higher than the return strategy based on the original route. RTN-Explorer can explore new areas while returning, but the return strategy based on

the original path can not explore new areas when returning. So the exploration efficiency of RTN-Explorer is significantly higher than that of returning based on the original path.

Comparison of Different Penalty Functions. As described in Sect. 4, We impose the minimum distance penalty proportional to the shortest distance from the UAV to the landing zone for UAV forced landing in the non-landing zone. Compared with the constant value penalty set in [17], our minimum distance penalty can significantly improve the return rate. Common distances also include European distance and Manhattan distance, which are relatively inexpensive to calculate. So in addition to the constant value penalty, we also compared the performance of the Euclidean distance penalty and the Manhattan distance penalty on the return rate (Fig. 4).

Since there is no correlation between the penalty function of landing in the non-landing area and the exploration rate, we only need to compare the return rate of these four penalty functions in different UAV numbers.

After a lot of experiments, we can get the following conclusion: the minimum distance penalty function we use can effectively improve the return rate. Using the Euclidean distance to the landing zone as the penalty can also obtain a high return rate. But the return rate of the Euclidean distance penalty function is lower than the minimum distance penalty function by at least 7.67%. The Manhattan distance penalty function yields a return rate of less than 60%. And the constant penalty yields a return rate of less than 11.88%.

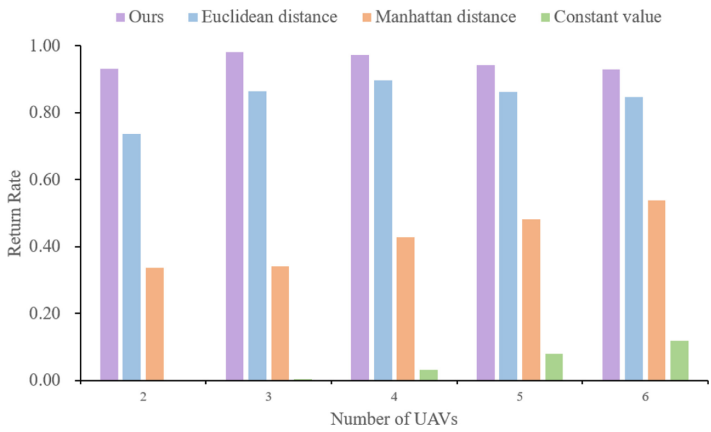


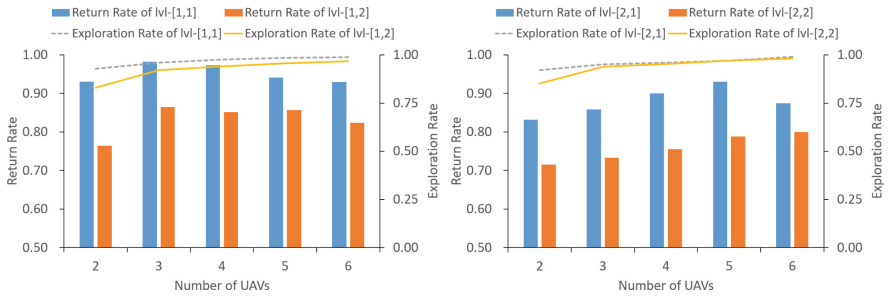
Fig. 4. Comparison of return rate under different penalty functions. The return rate of the method based on the constant value penalty function is 0.00% for two UAVs and 0.41% for three UAVs.

Comparison of Environments with Different Difficulties. At the beginning of each episode, we reinitialize the environment map. We allow the map generator to generate initial environment maps with different complexity according to the difficulty vector. Specifically, the difficulty vector consists of two elements $[d_m, d_t]$.

d_m represents morphological randomness, which can define the shape of obstacles in the environment. d_m controls the area each obstacle may occupy. The bigger the value of d_m, the larger the possible obstacle composite area in the training environment. d_m gets values from $\{1, 2\}$ discrete set.

d_t represents topological randomness, which defines the positions of obstacles on the map. The fundamental positions of the obstacles are equally arranged in a 3 columns - 3 rows format. d_t controls the deviation radius around these base positions. As the value of d_t increases, the topology of obstacles has more unstructured forms. d_t gets values from the $\{1, 2\}$ discrete set.

Higher values in the elements of the difficulty vector correspond to less structured behavior in the obstacles formation. As a result, a trained agent that has been successful in higher-difficulty training setups may have better generalization abilities. It can be seen from Fig. 5 that the environment of different difficulty levels has less impact on exploration rates. In contrast, the return rate is more sensitive to the difficulty of the environment. Nevertheless, when the difficulty vector $lvl = [2, 2]$, the exploration rate can still be maintained at more than 85%, and the return rate can be maintained at more than 70%.



(a) $d_m = 1$ (b) $d_m = 2$

Fig. 5. The sensitivity of the exploration rate and the return rate with respect to the different levels of the difficulty vector.

6 Conclusion

We propose an environment exploration strategy (RTN-Explorer) that satisfies the energy constraints. RTN-Explorer makes the UAV return to the landing zone before the energy is exhausted while ensuring exploration efficiency. Before training, we centralize, globalize, and localize environment map representations to improve performance. To satisfy the energy constraint, we design a penalty

function based on the shortest path distance to the landing zone and use the map generator to generate the environment map for training. A large number of simulation experiments show that RTN-Explorer is effective and robust. In order to further improve the stability of the system, we can increase the difficulty dimension in the map generator to increase the randomness of the environment. In future work, we'll expand the UAVs' action space to include altitude and continuous control, which will necessitate a different RL method than Q-learning, as well as adding height information to the agents' observations space.

References

1. Schuster, M.J., et al.: Towards autonomous planetary exploration. J. Intell. Robot. Syst. **93**(3), 461–494 (2019)
2. Hougen, D.F., et al.: A miniature robotic system for reconnaissance and surveillance. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 1, pp. 501–507. IEEE (2000)
3. Queralta, J.P., et al.: Collaborative multi-robot search and rescue: planning, coordination, perception, and active vision. IEEE Access **8**, 191617–191643 (2020)
4. Huang, Y., Cao, Z., Oh, S., Kattan, E., Hall, E.: Automatic operation for a robot lawn mower. In: Mobile Robots I, vol. 727, pp. 344–354. International Society for Optics and Photonics (1987)
5. Jager, M., Nebel, B.: Dynamic decentralized area partitioning for cooperating cleaning robots. In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), vol. 4, pp. 3577–3582. IEEE (2002)
6. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 1, pp. 476–481. IEEE (2000)
7. Banfi, J., Quattrini Li, A., Rekleitis, I., Amigoni, F., Basilico, N.: Strategies for coordinated multirobot exploration with recurrent connectivity constraints. Auton. Robot. **42**(4), 875–894 (2018)
8. Cao, C., Zhu, H., Choset, H., Zhang, J.: Tare: a hierarchical framework for efficiently exploring complex 3d environments. In: Robotics: Science and Systems Conference (RSS), Virtual (2021)
9. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
10. Niroui, F., Zhang, K., Kashino, Z., Nejat, G.: Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments. IEEE Robot. Autom. Lett. **4**(2), 610–617 (2019)
11. Koutras, D.I., Kapoutsis, A.C., Amanatiadis, A.A., Kosmatopoulos, E.B.: Marsexplorer: exploration of unknown terrains via deep reinforcement learning and procedurally generated environments. Electronics **10**(22), 2751 (2021)
12. Lee, W.C., Lim, M.C., Choi, H.L.: Extendable navigation network based reinforcement learning for indoor robot exploration. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 11508–11514. IEEE (2021)
13. Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F.: Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Trans. Veh. Technol. **69**(12), 14413–14423 (2020)

14. He, D., Feng, D., Jia, H., Liu, H.: Decentralized exploration of a structured environment based on multi-agent deep reinforcement learning. In: 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), pp. 172–179. IEEE (2020)
15. Geng, M., Xu, K., Zhou, X., Ding, B., Wang, H., Zhang, L.: Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration. Entropy **21**(3), 294 (2019)
16. Geng, M., Zhou, X., Ding, B., Wang, H., Zhang, L.: Learning to cooperate in decentralized multi-robot exploration of dynamic environments. In: Cheng, L., Leung, A.C.S., Ozawa, S. (eds.) ICONIP 2018. LNCS, vol. 11307, pp. 40–51. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04239-4_4
17. Theile, M., Bayerlein, H., Nai, R., Gesbert, D., Caccamo, M.: Uav coverage path planning under varying power constraints using deep reinforcement learning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1444–1449. IEEE (2020)
18. Zhou, B., Zhang, Y., Chen, X., Shen, S.: Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. IEEE Robot. Autom. Lett. **6**(2), 779–786 (2021)
19. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proceedings of the Second International Conference on Autonomous Agents, pp. 47–53 (1998)
20. Bautin, A., Simonin, O., Charpillet, F.: *MinPos*: a novel frontier allocation algorithm for multi-robot exploration. In: Su, C.-Y., Rakheja, S., Liu, H. (eds.) ICIRA 2012. LNCS (LNAI), vol. 7507, pp. 496–508. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33515-0_49
21. Lopez-Perez, J.J., Hernandez-Belmonte, U.H., Ramirez-Paredes, J.P., Contreras-Cruz, M.A., Ayala-Ramirez, V.: Distributed multirobot exploration based on scene partitioning and frontier selection. Mathematical Problems in Engineering 2018 (2018)
22. Amigoni, F., Banfi, J., Basilico, N.: Multirobot exploration of communication-restricted environments: a survey. IEEE Intell. Syst. **32**(6), 48–57 (2017)
23. Andre, T., Bettstetter, C.: Collaboration in multi-robot exploration: to meet or not to meet? J. Intell. Robot. Syst. **82**(2), 325–337 (2016)
24. Colares, R.G., Chaimowicz, L.: The next frontier: combining information gain and distance cost for decentralized multi-robot exploration. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 268–274 (2016)
25. Amato, C., Konidaris, G., Cruz, G., Maynor, C.A., How, J.P., Kaelbling, L.P.: Planning for decentralized control of multiple robots under uncertainty. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 1241–1248. IEEE (2015)
26. Murphy, R.R.: Human-robot interaction in rescue robotics. IEEE Trans. Syst. Man Cybern. Part C (Applications and Reviews) **34**(2), 138–153 (2004)
27. Oliehoek, F.A., Amato, C.: A concise introduction to decentralized POMDPs. Springer (2016)
28. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
29. Abadi, M., et al.: {TensorFlow}: a system for {Large-Scale} machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283 (2016)