Journal of *Marine Science and Engineering*

MDPI

Article

A Control Algorithm for Sea–Air Cooperative Observation Tasks Based on a Data-Driven Algorithm

Kai Hu [1,2], Xu Chen [1], Qingfeng Xia [3,4,5,*], Junlan Jin [1,2] and Liguo Weng [1,2]

1 School of Automation, Nanjing University of Information Science & Technology, Nanjing 210044, China; nuistpanda@163.com (K.H.); 20181223010@nuist.edu.cn (X.C.); 20201249090@nuist.edu.cn (J.J.); 002311@nuist.edu.cn (L.W.)
2 Jiangsu Provincial Collaborative Innovation Center for Atmospheric Environment and Equipment Technology, Nanjing University of Information Science & Technology, Nanjing 210044, China
3 School of Automation, Wuxi University, Wuxi 214105, China
4 School of Management and Engineering, Nanjing University, Nanjing 210093, China
5 State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
* Correspondence: fenix@jlxy.nju.edu.cn; Tel.: +86-18625186392

Abstract: There is tremendous demand for marine environmental observation, which requires the development of a multi-agent cooperative observation algorithm to guide Unmanned Surface Vehicles (USVs) and Unmanned Aerial Vehicles (UAVs) to observe isotherm data of the mesoscale vortex. The task include two steps: firstly, USVs search out the isotherm, navigate independently along the isotherm, and collect marine data; secondly, a UAV takes off, and in its one round trip, the UAV and USVs jointly perform the task of the UAV reading the observation data from USVs. In this paper, aiming at the first problem of the USV following the isotherm in an unknown environment, a data-driven Deep Deterministic Policy Gradient (DDPG) control algorithm is designed that allows USVs to navigate independently along isotherms in unknown environments. In addition, a hybrid cooperative control algorithm based on a multi-agent DDPG is adopted to solve the second problem, which enables USVs and a UAV to complete data reading tasks with the shortest flight distance of the UAV. The experimental simulation results show that the trained system can complete this tas, with good stability and accuracy.

Keywords: sea and air observation; multi-agent collaboration; data-driven; deep reinforcement learning

check for updates

Citation: Hu, K.; Chen, X.; Xia, Q.; Jin, J.; Weng, L. A Control Algorithm for Sea–Air Cooperative Observation Tasks Based on a Data-Driven Algorithm. *J. Mar. Sci. Eng.* **2021**, *9*, 1189. https://doi.org/10.3390/jmse9111189

Academic Editor: Kyung-Ae Park

Received: 16 September 2021
Accepted: 25 October 2021
Published: 27 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

A mesoscale eddy is a kind of marine phenomenon characterized by long-term closed circulation with a time scale from days to months and a spatial scale from tens of kilometers to hundreds of kilometers. It has a nonnegligible impact on weather prediction, marine chemistry, and the biological environment [1]. The current means of object-oriented observation are insufficient, so it is necessary to develop an automatic sea–air cooperative observation system to promote the study of mesoscale eddy-related air–sea interactions and their weather and climate effects. The system's primary goal is to realize the cooperative tasks of many kinds of intelligent equipment, such as USVs, UAVs, and so on. As shown in Figure 1, the observation of seawater isotherms will help us to understand the formation and propagation of mesoscale eddies. This observation includes two missions:

- Mission 1: USVs search for the isotherm of the mesoscale eddy, self-navigate along the isotherm, and collect ocean data;
- Mission 2: A UAV takes off, and in its one round trip, the UAV and USVs cooperatively move and the UAV reads all observation data from the USVs one by one with a limited distance.

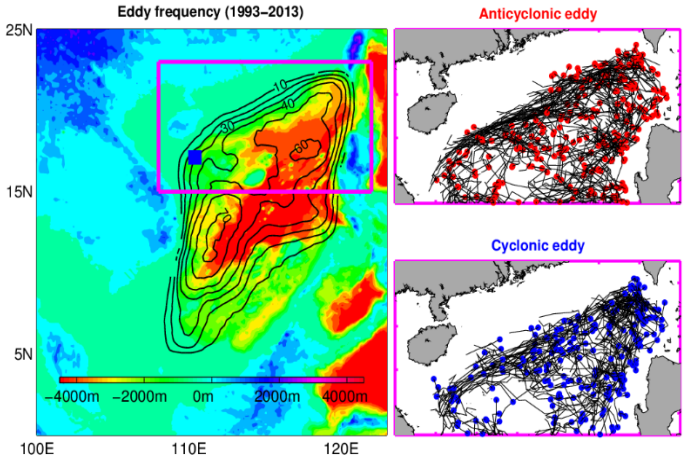


Figure 1. Statistical characteristics of mesoscale processes.

The ocean environment is dynamic and variable; thus, it is difficult to build an accurate environment and multi-agent system control model. Therefore, traditional control algorithms, which depend on model design, are difficult to build. In recent years, the rapid development of Reinforcement Learning (RL) and data-driven control theory has provided a new idea for solving the problem of multi-agent cooperative control.

For mission 1, in the field of USV and USV control, many outstanding researchers have designed USV schemes and control platforms [2–4]. Most studies are based on the ideas of path planning and trajectory tracking control to realize USV navigation, using GPS for remote path control and achieving a very accurate motion control effect. However, for mission 1 as stated in this paper, what we need is a scheme for the autonomous control of a USV in an unknown environment. This involves finding isotherms by autonomous decision-making and the control of the USV itself. Moreover, due to the "migration" movement of the mesoscale eddy itself, the water temperature in the same area also changes all the time, which leads to a very dynamic environment.

For mission 2, regarding the cooperative work between UAV and USVs, Polvara [5] introduced an end-to-end control technology based on deep reinforcement learning to land an Unmanned Aerial Vehicle on a visual marker located on the deck of a USV. At present, there is no relevant discussion about mission 2. However, whether we consider the tracking of the UAV in the unknown environment in task 1, or the cooperative control of the UAV and UAV in task 2, the main development direction of these two fields has pointed to the Reinforcement Learning scheme [6–10].

RL can find the optimal behavior from an unknown environment through the interaction between the system and its territory when the external environment does not provide a clear training signal. In 1990, Werbos [11] proposed adaptive dynamic programming based on RL. In 2014, Modares [12] proposed an online learning algorithm, Integral Reinforcement Learning (IRL), which is used to solve the optimal tracking control problem of linear or nonlinear systems whose system models are partially unknown. However, for multi-agent systems, the state of agents will change with time, and each agent is faced with emotional decision-making problems, which are uncertain and must be made in real-time. As a result, traditional decision-making methods such as RL are not suitable for solving continuous action control problems [13]. In recent years, Deep Reinforcement Learning (DRL) has received extensive attention [14–16]; it provides an optimal decision-making strategy for complex, high-dimensional multi-agent systems and can perform tasks effectively in challenging environments. Some classical DRL algorithms and their brief introductions are shown in Table 1.

Table 1. Summary of DRL algorithms.

Classification	Name	Introduction
Value-based RL	DQN	Two identical Q network structures are proposed, but they cannot deal with continuous action control.
	Double DQN [17]	The DQN training algorithm is improved to decouple the selection and evaluation of actions.
	Dueling DQN [18]	The DQN is improved, and the Q value is divided into the state value and the action dominance function.
Policy-based RL	PG [19]	The value-based approach is replaced with a policy-based approach.
Hybrid algorithm	AC [20]	A strategy network and value network are combined.
	A3C [21]	A general asynchronous and concurrent reinforcement learning framework is proposed, and the evaluation network is optimized.
	DDPG	The evaluation and policy network is used to update the model parameters iteratively, but it is not suitable for random environment scenarios.
	NAF [22]	A network is trained to output both an action and Q value simultaneously, but the algorithm is not mature.

The Deep Q Learning (DQN) algorithm is the pioneer of DRL, proposed by DeepMind [23] in 2013. It implements end-to-end learning from perception to action. Later, Wu [24] solved the path planning problem of single USV and multi-USV formation by using the DQN algorithm. Because the DQN algorithm selects an action by calculating the maximum Q value, the estimated value function is larger than the actual value function. With the expansion of the action space, the estimation error will continue to increase, and so the DQN algorithm cannot be applied to high-dimensional continuous motion control. In 2015, Lillicrap [25] proposed the Deep Deterministic Strategy Gradient (DDPG) algorithm based on the Actor–Critic (AC) framework. In 2019, Wang [26] realized the self-adjustment of a multi-AUV sliding mode controller at different speeds by using the DDPG algorithm. It is worth noting that the above control methods based on RL still have high requirements for the system model. With the application of digital sensor technology, people can obtain a great deal of data-carrying system information. Some researchers have proposed the use of these data to develop data-driven control protocols. The data can be used in the optimal control of a single agent and the cooperative control of multiple agents. In 2010, Xing [27] proposed an autonomous obstacle avoidance and trajectory tracking strategy based on data-driven control theory to solve the problem of multi-agent formation security. In 2019, Zhang [28] proposed an optimal formation control method based on a data-driven approach to solve the problem of multi-agent formation with virtual leaders. In 2017, Foerster [29] proposed a counterfactual multi-agent (COMA) policy gradient, which uses a counterfactual baseline that marginalizes out a single agent's action, while keeping the other agents' actions fixed. In 2019, Xiong [30] proposed a distributed data-driven control algorithm for multi-agent systems (MAS) based on a model-free adaptive control (MFAC) method.

Our contributions to achieving the engineering missions mentioned above in unknown, dynamic, and variable ocean environments are as follows:

1. Mission 1 requires the USV to follow the isotherm of the mesoscale eddy, which can be regarded as the navigation problem of the USV in an unknown environment. Because the state of the USV at sea will change with time, with the instability of this environment, it is difficult to understand and predict its strategy, which makes it challenging for the Q learning algorithm to converge. Unlike the supervised learning algorithm, which requires a large amount of sample data, the DDPG algorithm does not need an accurate mathematical model of the controlled object, which is of great significance for controlling a USV in unknown environments. In this scenario, the DDPG algorithm has certain advantages in dealing with high-dimensional contin-

uous state and action space problems. Therefore, we use the DDPG algorithm to learn these system data and train the data-driven DDPG controller of the related system to complete the task of the USV tracking isotherm.

2. Mission 2 requires multiple USVs and a UAV to be guided to work together and transmit a large amount of data. The control network of this kind of sea–air cross-domain heterogeneous system is very complex; in addition, under the condition of long distances at sea, the communication conditions are limited, so it is difficult to build an accurate environment and control model. Moreover, for the scenario of heterogeneous cooperation between drones, it is not a simple extension to train each agent separately. From the perspective of either the USV or UAV, the environment will become unstable. Because they update their respective strategies independently, it is difficult to guarantee the convergence of the algorithm. In view of the DDPG algorithm's deficiency in solving heterogeneous cooperative control, a cooperative control strategy of a heterogeneous multi-agent-based DDPG (MADDPG) is adopted in this paper.

The structure of this paper is as follows: the first section introduces the research background. The second section briefly analyzes the Markov decision process and the classical Q learning algorithm. In the third section, the USV tracking isotherm method based on data-driven and DRL approaches is proposed. In the fourth section, the sea–air cooperative observation method based on DRL is proposed. In the fifth section, the performances of the algorithms are verified by experimental simulation. The sixth section concludes our work.

2. Reinforcement Learning

One of the primary purposes of RL is to give full play to the interaction between the agent and the environment, take the environmental feedback and the agent's state as inputs, and learn knowledge more effectively by updating strategies in the ever-changing complex environment.

2.1. Markov Decision Process (MDP)

As shown in Figure 2, the MDP describes the learning environment, in which a goal can be learned via continuous interactions between an agent and the environment [6]. It can be represented as a four-element tuple:

$$M = [S, A, P, R] \tag{1}$$

where $S = \{s_1, s_2, \ldots, s_t\}$ represents the dynamic environment with a finite set of states, s_t denotes the state at time t, $A = \{a_1, a_2, \ldots, a_t\}$ represents actions executed by an agent, a_t denotes action at time t, R is the reward function, and P is the transition probability function, expressed as

$$P[s_{t+1} = s^{'} | s_t = s, a_t = a] \tag{2}$$

where s_{t+1} represents the next state after the execution of action a in the state s. The interaction between the agent and environment is shown as Figure 2. The agent is not only a learner but also a decision-maker—it selects action a_t with observed environment state s_t. The environment, in response to the actions taken by agent, updates its state to s_{t+1} and returns an immediate r_{t+1} to the agent.

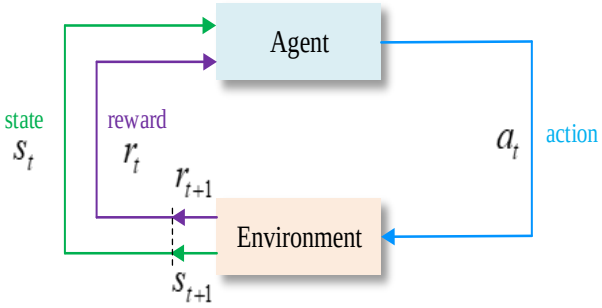


Figure 2. The agent–environment MDP interaction framework.

2.2. Q Learning

The aim of Q learning is to find an optimal control policy π^* for a given MDP [22]. π^* maximizes the action value function:

$$Q^*(s,a) = \max_{\pi}(E[\sum_{k=0}^{\infty}\gamma^k(r_{t+k}|s_t = s, a_t = a, \pi)]) \quad (3)$$

where $\gamma \in [0,1]$ is the discount factor that determines the return function. Q learning is an off-policy approach to solving MDPs and directly approximates the action value function via

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max(Q(s^{'},a) - Q(s,a))] \quad (4)$$

where $\alpha \in [0,1]$ is the learning rate. Note that the Q learning algorithm needs to initialize a Q table to record the expectation of the action value function, making it only suitable for dealing with simple situations with a small action and state space. However, most problems in the real world require a large number of states and action spaces, which makes it unrealistic to build a Q table containing all states and actions.

3. USV Tracking Isotherm Algorithm

3.1. Deep Deterministic Policy Gradient Algorithm

While Q-Learning or DQN solves problems with high-dimensional observation spaces, it can only handle discrete and low-dimensional action spaces. Many tasks of interest, most notably physical control tasks, have continuous (real valued) and high-dimensional action spaces. Q-Learning or DQN cannot be straightforwardly applied to continuous domains since it relies on a finding the action that maximizes the action-value function, which in the continuous valued case requires an iterative optimization process at every step. Lillicrap [25] proposed the DDPG algorithm, which combines AC with insights from DQN; it can learn competitive policies for all of our tasks using low-dimensional observations (e.g., cartesian coordinates or joint angles) using the same hyper-parameters and network structure. A key feature of the approach is its simplicity: it requires only a straightforward Actor–Critic architecture and is a learning algorithm with very few "moving parts", making it easy to implement and scale to more difficult problems and larger networks. It has achieved good performance in the track optimization of USVs, flight control of UAVs, cooperative navigation of multi-mobile robots, and so on [8–10].

As shown in Figure 3, in the DDPG algorithm, depth neural networks with parameters θ^{μ} and θ^{Q} are used to represent the deterministic strategy $a = \pi(s|\theta^{\mu})$ and action value function $Q(s,a|\theta^{Q})$. Among them, the policy network corresponds to the Actor in the AC framework to update the policy. The value network approximates the value function corresponding to the state action and provides relevant gradient information, which corresponds to the Critic in the AC framework.

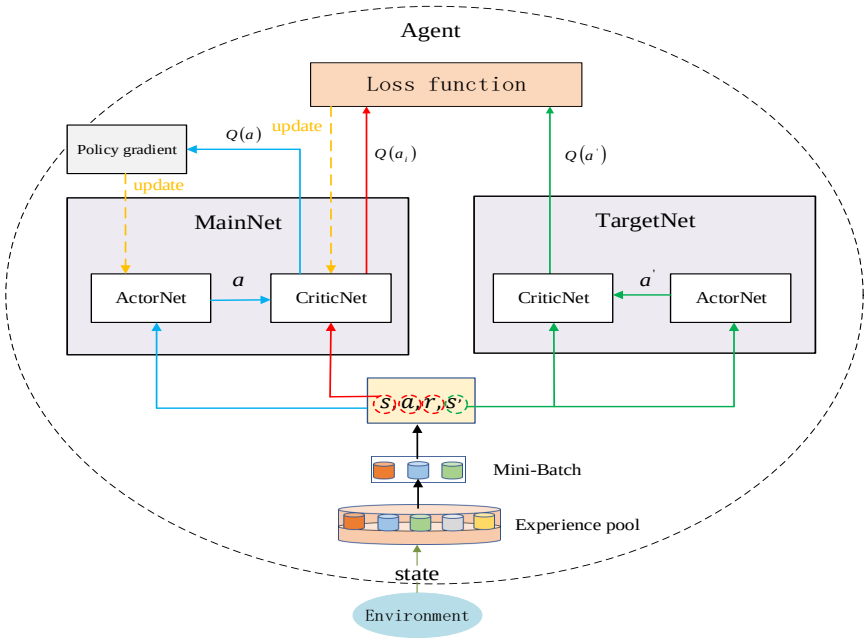


Figure 3. DDPG network training process.

Firstly, the objective function is defined as the total return with discount, shown as

$$J(\theta^{\mu}) = E_{\theta^{\mu}}[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^n r_n] \tag{5}$$

where J is used to measure the performance of a strategy μ, and θ^{μ} is the parameter of the policy network. A CNN is used to approximate the μ function, which is called the policy network. $E_{\theta^{\mu}}[\cdot]$ represents the expected value of the total return.

Secondly, the objective function is optimized end-to-end by a random gradient method to improve the total return, the gradient of the objective function with respect to θ^{μ} is equivalent to the expected gradient of the Q-valued function with respect to θ^{μ}:

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{\mu}} = E_s[\frac{\partial Q(s,a|\theta^{Q})}{\partial \theta^{\mu}}] \tag{6}$$

where θ^{μ} is the parameter of the Q network. A CNN is used to simulate the Q function and called a Q network.

Then, according to the deterministic strategy $a = \pi(s|\theta^{\mu})$, we obtain

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{\mu}} = E_s[\frac{\partial Q(s,a|\theta^{Q})}{\partial a}\frac{\partial \pi(s|\theta^{\mu})}{\partial \theta^{\mu}}] \tag{7}$$

where $\pi(s|\theta^{\mu})$ represents the conditional probability density related to the strategy. $Q(s,a|\theta^{Q})$ represents the parameter probability distribution, which randomly selects an action according to the parameter vector θ^{Q}. The update of the policy network parameters can be completed by raising the Q value.

Finally, as with the DQN algorithm, the DDPG algorithm updates the evaluation network by updating the value network, and its gradient information is expressed as follows:

$$\frac{\partial J(\theta^{\mu})}{\partial \theta^{Q}} = E_{s,a,r,s' \sim D}[TargetQ - Q(s,a|\theta^{Q})\frac{\partial Q(s,a|\theta^{Q})}{\partial \theta^{\mu}}] \tag{8}$$

$$TargetQ = r + \gamma Q^{'}(s^{'}, \pi(s^{'}|\theta^{\mu^{'}})|\theta^{Q^{'}}) \tag{9}$$

where D represents the experience playback pool and $TargetQ$ represents the target Q value. $\theta(\mu^{'})$ is the target policy network parameter copied by θ^{μ}, and $\theta(Q^{'})$ is the target Q network parameter copied by θ^{Q}, which updates the value network by gradient descent.

Because the target policy is a deterministic strategy, in contrast to the execution strategy, the evaluation of the value function of the DDPG algorithm adopts the Q-learning approach with different strategies. More specifically, the loss function (TD error) of the online Q network is calculated as follows:

$$L = \frac{1}{N}\sum\left(y_i - Q\left(s_i, a_i\middle|\theta^{Q}\right)\right)^2 \tag{10}$$

where y_i is the "label" obtained by the target policy network and the target Q network, which is defined as follows:

$$y_i = r_i + \lambda Q^{'}\left(s_{i+1}, \mu^{'}\left(s_{i+1}\middle|\theta^{\mu^{'}}\right)\middle|\theta^{Q^{'}}\right) \tag{11}$$

The data sample $(s, a, r, s^{'})$ in the process of interaction between the agent and the environment is stored in the experience pool, and a small sample is randomly selected from the sample pool for neural network training, which breaks the relationship between the samples. At the same time, because the update of the Q-value function directly leads to unstable training, a separate target network is set up in DDPG to evaluate TD deviation. Finally, DDPG updates the network to the target network as follows:

$$\begin{cases} \theta^{Q^{'}} \leftarrow \delta\theta^{Q} + (1-\delta)\theta^{Q^{'}} \\ \theta^{\mu^{'}} \leftarrow \delta\theta^{\mu} + (1-\delta)\theta^{\mu^{'}} \end{cases} \tag{12}$$

where δ is the update coefficient, and its value is generally small (0.1 or 0.01).

3.2. USV Tracking Isotherm Algorithm

As we can see from Figure 1, the spatial scale of mesoscale eddies ranges from tens of kilometers to hundreds of kilometers. In our ideal observation task, satellite, several USVs and UAVs are used. Firstly, a satellite is used to roughly observe the formation of the mesoscale and find its center and boundary. Then, our mother ship drives to the center and places USVs, which drive to the mesoscale boundary with a cross formation (shown in Figure 4). After collecting data all the way, we can find points with the most largest temperature change. Finally, USVs drive to these points and begin to execute our algorithm to find isotherms.

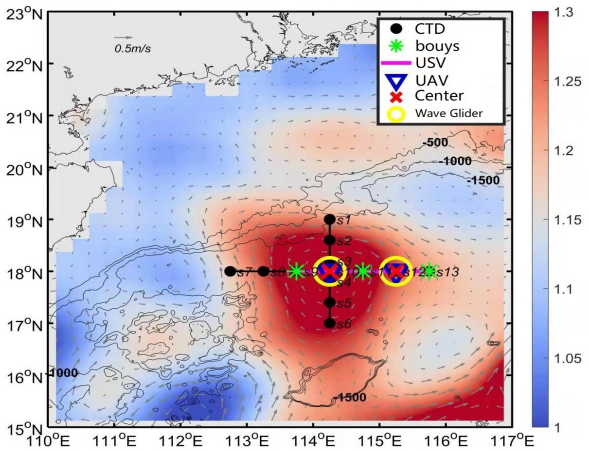


Figure 4. Finding start positions.

This paper uses three USVs as a demonstration. According to the observation mission plan (shown in Figure 5), Task 1 needs USV_1, USV_2, and USV_3 to be guided to search for isotherms in their respective areas and navigate independently along the isotherms. If the isotherm is regarded as the target point or target trajectory, this mission can be regarded as the navigation problem of an unmanned vehicle in an unknown environment.

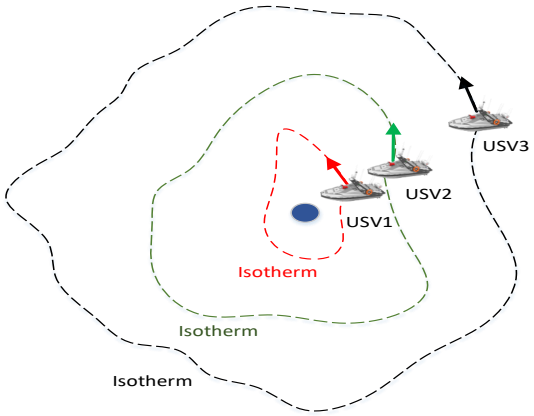


Figure 5. Multiple USVs cooperate to search isotherms.

The principle block diagram of DDPG for USV tracking isotherm control is shown in Figure 6.

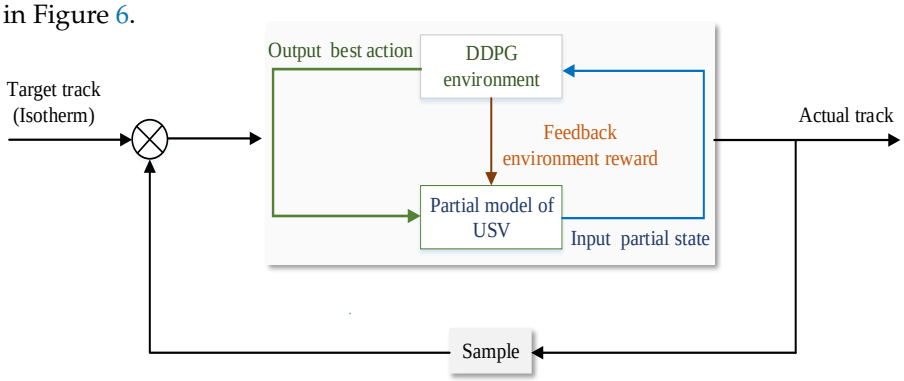


Figure 6. Principle block diagram of USV tracking isotherm.

(1) MDP for USV tracking isotherm

The Markov decision model includes the design of the state space, action space, and return function.

State: The state space is defined as the USV's position coordinate (x, y) and heading direction φ. As a result, the angle θ_t between the USV and the target trajectory can be calculated and normalized to $[-1, 1]$. In addition, the state space also includes the environmental state information returned by the temperature sensor. The temperature sensor collects the water temperature data K_t of the current position and normalizes it to $[0, 1]$. According to the change of water temperature, we can judge whether a USV is sailing at the desired isotherm. The definition of state space is as follows:

$$s = \{\theta_t, K_t\} \tag{13}$$

Action: Considering the motion characteristics of the USV, it is assumed that the line speed of the USV remains constant during the voyage. In contrast to most existing path planning methods for USVs using reinforcement learning, this paper defines a fine action space, and the formula is as follows:

$$a = \{a_1, a_2, \cdots, a_{13}, back\} \tag{14}$$

where the action space a is a vector containing 14 elements, and $a_1 \sim a_{13}$ are angle values ranging from $-60^\circ(a_1)$ to $60^\circ(a_{13})$ with increments of 10° (the range of this value can be changed according to the dynamic characteristics of the unmanned vehicle), and it is normalized to $[-1, 1]$. *back* indicates a return to the nearest situation where reward is 1—this is a remedial strategy, similar to a Windows system restart, to prevent a USV from getting lost when it encounters an isotherm with a large corner angle. This action will be executed when a series of reward values are below a threshold.

In essence, this design depends on the characteristics of the mesoscale eddy. As we can see from Figure 4, for a warm eddy, the hotter it is toward the center, the colder it is away from the center. For a cold eddy, the opposite would be true. Thus, for a USV, when it (in fact, our algorithm) knows whether it us moving clockwise or counterclockwise in a warm or cold eddy, this action space is comparably easy to design and learn.

In addition, according to the defined state and action spaces, the state transition with a given action a_i by adhering to the kinematics of a USV can be calculated as

$$x^{'} = x + x\cos\varphi^{'} \tag{15}$$

$$y^{'} = y + y\sin\varphi^{'} \tag{16}$$

$$\varphi^{'} = \varphi + a_i \tag{17}$$

where $s = [x, y, \varphi]$ is the state in the current step, $s^{'} = [x^{'}, y^{'}, \varphi^{'}]$ is the state in the next step, and a_i is the chosen action in the current step.

Reward: To generate the shortest path to the target trajectory (isotherm)—that is, to ensure that each step of the action can bring the USV close to the isotherm—in this paper, the following set of return functions are designed:

$$r = \begin{cases} r_{arrive}, if \quad |K_t - K_0| \le K_{goal} \\ r_{direction}, others \end{cases} \tag{18}$$

where $r_{arrive} = 1$ is the terminal reward function. Taking USV_1 as an example, K_0 represents the temperature collected by USV_1 in the starting point, and K_t represents the water temperature where USV_1 is located at the current time. If the absolute difference between K_t and K_0 is less than or equal to the threshold T_{goal}, then USV_1 is within the isotherm search range and activates the function. At the same time, in order to guide USV_1 to sail continuously toward the target trajectory, a steering function is defined as

$$r_{direction} = w_1(\theta_t - \theta_{t-1}) - w_2|K_t - K_{t-1}| \tag{19}$$

where K_t and K_{t-1} represent the water temperature in the area where the USV is located at time t and $t-1$, and θ_t represents the angle between the USV heading and the target trajectory at time t. w_1 represents the reward coefficient and w_2 represents the punishment coefficient, both of which are adjusted according to the parameter adjustment process. If the previous action moves the USV away from the isotherm, a penalty is given; if the previous action brings the USV close to the isotherm, a certain reward is given. This means that the USV is not sparsely rewarded in the process of exploration and accelerates the convergence speed of the algorithm. In addition, the distance between the USVs is also considerable, so there is no need to consider the problem of avoiding obstacles and colliding with each other.

(2) Design of network structure

As shown in Figure 7, the policy network uses a CNN with one input and one output. The network's input is the environmental state information; that is, the water temperature information and the current motion state information of the USV. The output is the action instruction. The evaluation network uses a CNN with two inputs and one output. The network's input includes the action of the environment state information and policy network's output. The evaluation network output is the evaluation index of the

current strategy; that is, the Q value of the action. At the same time, the action matrix is also input into the evaluation network. Finally, the third layer neuron of the network input by the state–space matrix is merged with the second layer neuron node of the action matrix input network, and the output value is obtained through the ReLU activation function.

There will be many temperature acquisitions in one action decision-making round. In essence, this action decision-making is mainly based on the prior knowledge generated by temperature change trends and mesoscale eddy characteristics mentioned above.

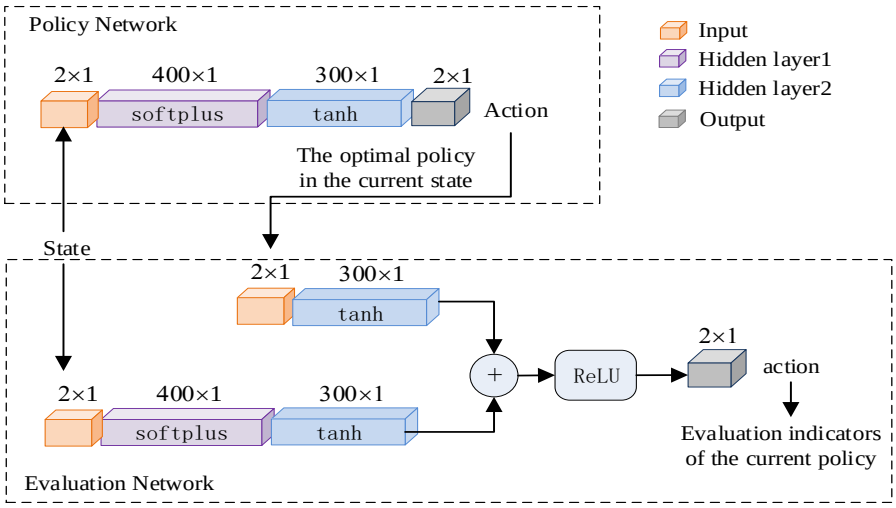


Figure 7. Design of strategy network and evaluation network structure.

4. Sea–Air Cooperative Control Algorithm

4.1. Multi-Agent Deterministic Policy Gradient Algorithm (MADDPG)

The distance between maritime agents is very long; the Beidou system can communicate, but its bandwidth is very small, at only 77 bytes per minute, so the Beidou system can only used for the transmission of limited control data. An ordinary radio system cannot realize long-distance communication, but it has a high bandwidth in a short distance. Therefore, we use Beidou to control the agents' movement and allow the UAV to read a large amount of data from the USVs in a short distance. However, even so, the bandwidth of Beidou control is very small, and agents only can communicate once in more than 10 min. Therefore, we face the problem of sea–air cooperative control under the condition of limited communication.

Mission 2 is to conduct the rendezvous of multiple USVs with a UAV in a specific time and domain so that the UAV can collect data by flying one round per day. As shown in Figure 8, taking three USVs and one UAV as an example, the main task of this mission is to guide USV_1, USV_2, and USV_3 to adjust their current speed based on their distance and direction to rendezvous with the UAV at different times. At the same time, the UAV should adjust its speed according to the position information of USVs. It is hoped that the result of control can be finished in the UAV's one round trip. In summary, mission 2 can be regarded as the problem of multiple USVs cooperating to track multiple target points (UAVs at different times); that is, sea–air heterogeneous cooperative control.

The scenario of heterogeneous cooperation between drones is not a simple extension of training each agent separately; in this case, whether from the perspective of the UAV or USVs, the environment will become unstable. Because they update their respective strategies independently, traditional RL cannot guarantee the algorithm's convergence, and the policy-based algorithm will also increase the number of the USVs, which aggravates the problem and causes more considerable variance. In summary, the DDPG algorithm is no longer suitable for sea–air heterogeneous cooperative control scenarios. Of course, through the modeling of the environment and agents, some researchers use the model-based strategy method; that is, updating the learning strategy according to the gradient

backpropagation in deep learning or predicting the next state of the environment based on certain assumptions and communication with each other, but this does not apply to situations in which maritime communication is limited. The environment is limited, and the cost is too high. It does not meet the requirements of developing a sea–air interface networking observation system with high precision, low cost, low risk, and automation.

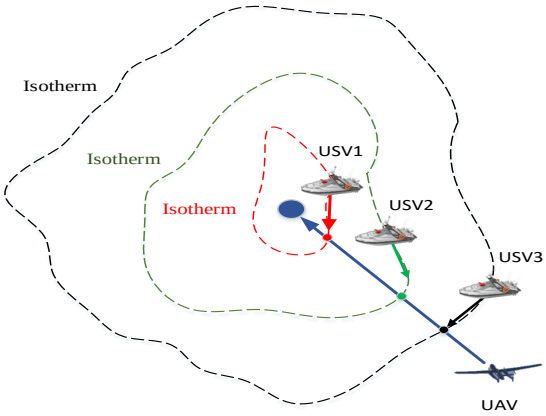


Figure 8. Heterogeneous cooperation between USVs and a UAV.

As communication between agents at sea is limited and we need a system structure of centralized training and distributed execution, we have to choose an algorithm from MADDPG and its few derived algorithms. COMA [29] is one of the famous algorithms derived from MADDPG; it uses a global evaluation function to evaluate all current actions and states, which improves the efficiency of information sharing in training and the cooperation ability between agents, but it cannot be used in a continuous action space like MADDPG. Furthermore, considering the algorithm based on classical architecture, we can easily verify whether the performance of the main MADDPG algorithm is feasible in our project; thus, to facilitate further improvement in the future, we adopt the MADDPG algorithm in mission 2.

In detail, we use θ to represent the parameter set of all agents:

$$\theta = \{\theta_1, \theta_2, \cdots, \theta_n\} \tag{20}$$

We use π to represent the policy set of all agents:

$$\pi = \{\pi_1, \pi_2, \cdots, \pi_n\} \tag{21}$$

A Markov game composed of multiple agents is a series of states configured by all agents; that is, the global environmental state (observed), which is expressed as

$$s = \{o_1, o_2, \cdots, o_n\} \tag{22}$$

where s represents the global observation, and o_i represents the observation of the i-th agent; that is, the state. Then, the cumulative expected return of the i-th agent at time t is

$$J(\theta_i) = \sum_{s\sim\rho^{\pi}, a_i\sim\pi_i} \left[t = \sum_{t=0}^{\infty} \gamma^t r(i,t) \right] \tag{23}$$

where $J()$ represents the total reward, $\rho^{\pi}(s)$ represents the discount status distribution, and γ is the discount factor, which is used to calculate the expectation of cumulative return in the future. Then, the corresponding random policy gradient is

$$\nabla_{\theta_i} J(\theta_i) = E_{s\sim\rho^{\pi}, a_i\sim\pi_i} \left[\nabla_{\theta_i} \log \pi_i(a_i|o_n) {\theta_i}^{\pi}(s, a_1, \cdots, a_n) \right] \tag{24}$$

where $E_{s\sim\rho^{\pi}}[]$ represents the expected value of the discounted state distribution $\rho(s)$, and $\nabla_{\theta_i}\log\pi_i(a_i|o_n)$ is the score function. $\pi_i(a_i|o_n)$ is a conditional probability distribution in state o_n for each action a_i. ${\theta_i}^{\pi}(s,a_1,\cdots,a_n)$ represents the state action-value function of the i-th agent, and the output of this value is obtained by taking the actions of all agents and the global environment state as inputs. This can be extended to the deterministic policy μ, which represents the deterministic policy set of all agents:

$$\mu = \{\mu_1,\mu_2,\cdots,\mu_n\} \tag{25}$$

The gradient formula of its deterministic strategy is as follows:

$$\nabla_{\theta_i}J(\theta_i) = E_{s,a\sim D}\left[\nabla_{\theta_i}\mu_i(a_i|o_i)\nabla_{a_i}{\theta_i}^{\mu}(s,a_1,\cdots,a_n)\Big|_{a_i=\mu_i(o_i)}\right] \tag{26}$$

where D represents the experience playback pool, which is used to store the empirical data of all agents interacting with the environment. Each piece of empirical data is made up of a set of $\left(s,s^{'},a_1,\cdots,a_n,r_1,\cdots,r_n\right)$, and the set of target strategies is represented as follows:

$$\mu^{'} = \left\{{\mu_1}^{'},{\mu_2}^{'},\cdots,{\mu_n}^{'}\right\} \tag{27}$$

Under the centralized training mode, MADDPG still uses the idea of a time difference and the idea of a target network to update the evaluation network parameters. If the target network contains parameter ${Q_i}^{'}$ with lag update, the objective function used to evaluate the network approximation is

$$y = r_i + \gamma {Q_i}^{\mu^{'}}\left(s^{'},{a_2}^{'},\cdots,{a_n}^{'}\right)\Big|_{{a'}_j={\mu'}_j(o_j)} \tag{28}$$

The loss function (TD error) is calculated from the above formula:

$$L(\theta_i) = E_{s,a,r,s'}\left[\left({Q_i}^{\mu}(s,a_1,\cdots a_n) - y\right)^2\right] \tag{29}$$

where ${Q_i}^{\mu}$ represents the target network, and $s^{'}$ represents the next state after the execution of action a in state s.

Obviously, it is necessary to obtain the action strategies of all agents in the environment due to the approximation of the global state action value function. Therefore, it is necessary to obtain the action output value of each agent in the process of centralized training, but the communication between agents in many scenarios is limited, so centralized training can also be achieved by estimating the strategies of other agents.

In order to improve the stability of the algorithm and let each agent learn multiple strategies, the overall effect of all strategies is used to improve the stability of the algorithm. Specifically, the strategy μ_i of the i-th agent is composed of a set of k sub-policies. If only one sub-strategy ${\mu_i}^{(k)}$ is used in each round of training, the overall return of the policy set is maximized as follows:

$$J_e(\mu_i) = E_{k\sim unif(1,K),s\sim\rho^{\mu},a\sim{\mu_i}^{(k)}}[R_i(s,a)] \tag{30}$$

where $unif(1,K)$ is a set of values uniformly distributed from 1 to K, and a set of sub-empirical playback pools ${D_i}^{(k)}$ is constructed for each sub-strategy k. To optimize the overall effect of the policy set, the update gradient for each sub-policy is as follows:

$$\nabla_{{\theta_i}^{(k)}}J_e(\mu_i) = \frac{1}{K}E_{s,a\sim{D_i}^{k}}\left[\nabla_{{\theta_i}^{(k)}}{\mu_i}^{(k)}(a_i|o_i)\nabla_{a_i}{\theta_i}^{\mu_i}(s,a_1,\cdots,a_n)\Big|_{a_i={\mu_i}^{(k)}(o_i)}\right] \tag{31}$$

where $\nabla_{a_i}{\theta_i}^{\mu_i}(s,a_1,\cdots,a_n)$ represents the gradient of the action-value function, which includes the actions of all agents $a=\{a_1,\cdots a_n\}$ and the environmental state quantity s. o_i

represents the observation of the i-th agent. Generally speaking, s includes the status of all agents and some additional information. Taking the convergence of drones as an example, the input of the evaluation network is the status of all drones and their positions, and the output $\theta_i^{\mu_i}(s, a)$ is returned to the strategic network to evaluate the actions performed by the actors.

In summary, the MADDPG algorithm is essentially an extension of the DDPG algorithm, which has the following characteristics: first, the optimal strategy is obtained by autonomous learning and can output the optimal global action by knowing the local information; secondly, the MADDPG algorithm does not need to know the dynamic model of the environment, and the algorithm can be applied not only to multi-agent cooperation but also to multi-agent competition.

4.2. Sea–Air Cooperative Control Algorithm

According to the MADDPG in the previous section, the environmental state of the confluence of multiple USVs and a UAV is defined. Before that, it is first necessary to determine the environment status of a single USV and UAV:

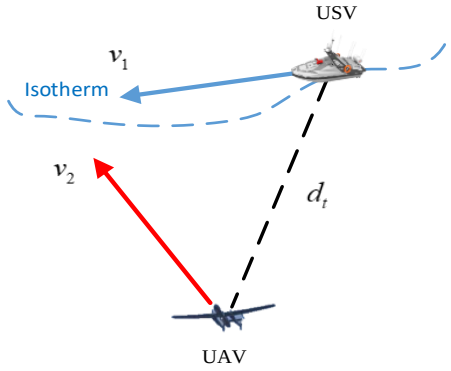


Figure 9. Schematic diagram of the environmental status of a single USV and UAV.

Taking USV_1 as an example, to achieve the convergence of USV_1 and the UAV within a set time domain, the relative distance between the USV and UAV needs to be considered (as shown in Figure 9). Assuming that the position coordinate of UAV at the current time is (x_{UAV}, y_{UAV}) and the position coordinate of USV is (x_{USV1}, y_{USV1}), then the distance between the two at time t is

$$d_t(USV_1, UAV) = \sqrt{(x_{UAV} - x_{USV1})^2 + (y_{UAV} - y_{USV1})^2} \tag{32}$$

According to the sea–air cooperative observation mission scenario in Section 2, three USVs move along different isotherms, respectively, and their trajectories will not be changed by the UAV. The UAV's velocity can be adjusted in the range of 3∼10 knots, where the most energy-saving speed is 5 knots. At the same time, UAV adopts compound wings—it can perform vertical takeoff and landing, but due to the limited energy of UAV, its maximum theoretical distance is 200 km, which limits the heading of the UAV to be fixed; it always flies to the center of the mesoscale eddy from the mother ship located outside eddy. After reaching the maximum distance of 100 km, it turns back to mother ship 1 or goes straight to mother ship 2 or an island, and the linear velocity can only be adjusted in the range of 0∼100 km/h. This is because the changes of linear acceleration and angular acceleration will consume energy from the UAV, which is not conducive to a UAV completing the collection of all the observed data by USVs in one flight.

The environment states of the three USVs and the UAV are defined as

$$S = \{d_t(USV_1, UAV), d_t(USV_2, UAV), d_t(USV_3, UAV)\} \tag{33}$$

The action space controlled by three USVs and the UAV is defined as

$$A = \left\{ {a_t}^{\mathrm{USV}_1}, {a_t}^{\mathrm{USV}_2}, {a_t}^{\mathrm{USV}_3}, {a_t}^{\mathrm{UAV}} \right\} \tag{34}$$

where ${a_t}^{\mathrm{USV}_1}$ and ${a_t}^{\mathrm{UAV}}$ represent the linear velocities of USV_1 and the UAV. Then, according to the scenario of the sea–air cooperative observation mission, the USV needs to rendezvous with the UAV within a specified time domain. Therefore, when setting the reward function of collaborative control for a single USV and UAV, time, distance, and energy factors need to be taken into account:

$$r = \begin{cases} r_{converge}, if \quad d_t(\mathrm{USV}_i, \mathrm{UAV}) \le d_{\max} \\ r_{distance}, others \end{cases} \tag{35}$$

where $r_{converge}$ represents the rendezvous reward function, which is a positive value. One of the key standards for the successful convergence of a USV and UAV is to complete the task of the "reading" of observation data. $d_{\max}$ means the maximum distance at which a drone can collect observation data from a drone. When the distance ${d_t}^{\mathrm{USV}_1}$ between USV and UAV is less than or equal to $d_{\max}$, the confluence is considered successful and the function is activated. $r_{distance}$ stands for the distance reward function, which is used to guide the USV and UAV to converge with the shortest time and minimum energy loss. The smaller the distance between USV and UAV, the greater the rewards. The expression of the distance reward function is as follows:

$$r_{distance} = -\lambda_1 (d_t - d_{t-1}) - \lambda_2 \left(\left| {v_t}^{\mathrm{UAV}} - {v_{t-1}}^{\mathrm{UAV}} \right| \right) \tag{36}$$

where λ_1 represents the distance reward factor, d_t represents the distance between the USV and UAV obtained by the current sample, and d_{t-1} represents the distance at the previous step. A difference is found between d_{t-1} and upper d_t; if $(d_{t-1} - d_t)$ is positive, it means that the distance between USV and UAV is decreasing, and the action of the previous step brings them closer, thus giving them a certain reward; otherwise, the action of the previous step is moving them further away, and thus a certain punishment is given. λ_2 represents the penalty coefficient and ${v_t}^{\mathrm{UAV}}$ represents the linear velocity at time t. When the last moment and the current linear speed change too much, a certain punishment is given. This is because the change of the UAV's linear speed will lead to excessive energy loss, which is not conducive to the UAV meeting the maximum range flight requirements.

Finally, when defining the global reward function, it is necessary to consider that the collaborative algorithm based on MADDPG should not only input the actions of the policy network output of USV_1 but also input the output of the USV_2 and USV_3 policy networks to form a combined state action evaluation network. Therefore, in order to realize the cooperative control of multiple USVs and a UAV, the reward function should be adjusted accordingly. After the execution of the action reaches a new state, the distances between USV_1, USV_2, USV_3, and the UAV are calculated in turn, which are used as the standard for evaluating rewards and punishments. The purpose of the collaborative algorithm is to enable all USVs to successfully converge with the UAV. The expression of the global reward function is as follows:

$$r = \begin{cases} 1, if \quad d_t(\mathrm{USV}_i, \mathrm{UAV}) \le d_{\max} \\ -\lambda_1 \sum_i^{i=Nt} (d(\mathrm{UAV}, \mathrm{USV}_i)) - \lambda_2 \left(v_t^{\mathrm{UAV}} - v_{t-1}^{\mathrm{UAV}} \right), others \end{cases} \tag{37}$$

where Nt indicates the total number of USVs participating in the networking at this stage, which is less than or equal to 3. Whenever a USV completes the communication task, it will be excluded from the networking task. At the same time, $\lambda_2 \left(v_t^{\mathrm{UAV}} - v_{t-1}^{\mathrm{UAV}} \right)$ means that the change of speed of the UAV will lead to a loss of energy, and when the change of the linear velocity of the UAV is small, the relative penalty value is small.

In addition, as can be seen from Figure 8, USV_1, USV_2, and USV_3 are controlled by three separate processes. When USV_3, which is located on the outermost isotherm, first completes the rendezvous with UAV, this first step is easy to realize, because the takeoff time of UAV is controllable, and the initial state of this control task is conducive to the realization. Then, USV_1 and USV_2 continue to carry out the rendezvous task; moreover, they can finish the mission when the UAV turns back. In other words, there are two opportunities for USV_1 and USV_2.

For the reward function, to achieve the global goal, each USV should not consider its mission in isolation. Therefore, in addition to reaching the reward, the single-step return function of a single USV during exercise is adjusted to all USVs' related positions related to the distance to the UAV. According to formula (37), the distance reward function $r_{distance}$ is summed up to three USVs. The closer the three USVs are to the UAV, the greater the global reward value.

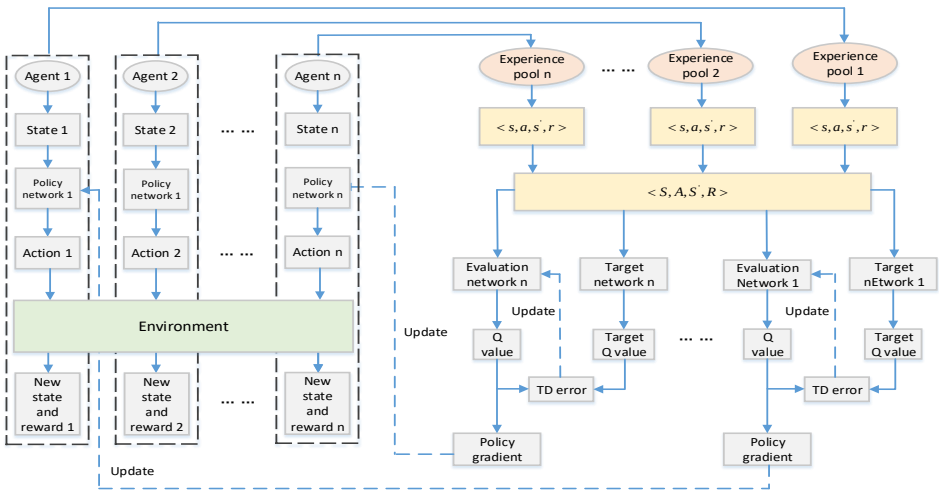


Figure 10. The framework of MADDPG algorithm.

The framework of the MADDPG algorithm is shown in Figure 10. For each agent, its state s_i is inputted into its policy network, and the action a_i is obtained. The action is executed in the simulation environment to reach the new state $s_i^{'}$ and get a reward r_i. At the same time, the interactive data are stored in their exclusive experience pool D_i in the form of a tuple $\left\langle s_i, a_i, s_i^{'}, r_i \right\rangle$. In the new state $s_i^{'}$, the agent will continue to interact with the environment according to its current policy network and continue to generate data. Thus, other agents also interact with the environment and store experience inputs in their own experience pools. When updating the network, we take agent i as an example: first, according to the size of the batch, a series of random numbers are generated with the capacity of the experience pool as the maximum value. Then, a batch of data at the same time is sampled from the experience pool of all robots with random numbers as the index, and a batch of tuples $\left\langle S, A, S^{'}, R \right\rangle$ is obtained by splicing them. The S and $S^{'}$ values of each tuple are the combined state of all agents at the same time, and A is the combined action of all agents. Reward R only relates to the reward value of agent i. Next, $S^{'}$ is input into the target policy network i to get action A. Then, $S^{'}$ and $A^{'}$ are jointly inputted into the target evaluation network i to obtain the estimated target Q value for the next time. Then, the evaluation network is used to obtain the actual Q value, the TD deviation is used to update the evaluation network, and the policy gradient of the Q value is used to update the policy network. Other agents update the network, and thus the main difference is input.

5. Simulation Results and Analysis

This section provides a detailed analysis and discussion of the simulation results of thhe proposed algorithms. The experimental contents mainly include the following:

1. The experimental verification of the USV tracking isotherm algorithm;
2. The experimental verification of the sea–air cooperative control algorithm.

The hardware configurations used as follows: one Intel i7-9700K CPU, two-way Nvidia Geforce GTX2080Ti graphics cards, and 4 × 16 Gb memory. Algorithms were programmed in the Python language, and the deep learning framework was TensorFlowgpul.4.0.

5.1. Experiment on USV Tracking Isotherm Algorithm

Firstly, the input, output, and return functions of the single USV tracking isotherm experiment were defined by the Markov decision model in Section 2.2 and the design of the DDPG algorithm network. Among them, the input parameters included the navigation information of the current USV and the water temperature information measured by the temperature sensor. The output parameter was the control instruction of the USV. The reward function included reaching the target track (isotherm), the reward of being close to the target track, and the punishment of being far away from the target track.

Table 2. Parameter setting of DDPG algorithm.

Parameters	Value
Number of training rounds max-el	1000
Learning rate of the policy network l_{r_a}	0.0001
Learning rate of the evaluation network l_{r_c}	0.001
Discount factor γ	0.99
Capacity of experience pool R-size	10,000
Update parameters of the target network δ	0.01
Coefficient of reward function w_1, w_2	0.5, 0.5

Next, we set the parameters related to the algorithm (as shown in Table 2). Because of the unpredictable shape of the mesoscale eddy isotherm, it is difficult to build the environmental model.The elliptical isotherm is designed as the target trajectory, and the DDPG algorithm is used to guide the USV to move along the target trajectory. As shown in Figure 11, the blue dotted line represents the target trajectory, the temperature is 10 °C and the red dot indicates the starting position of the USV. The focal coordinate of the elliptical trajectory is set to (300, 200), the temperature is 15 °C, the temperature gradually drops from 15 °C in the center to 10 °C on the trajectory, and the lengths of the major and minor axes are 300 and 150, respectively. The starting point coordinates of the USV are (220, 20). The USV moves counterclockwise along the fitted isotherm, the rate of temperature sensor acquisition is set to 10 times per second, and the linear velocity remains constant at 0.15 m/s—almost 3 knot—only changing the angular velocity to adjust the course of the USV. At the same time, to verify the algorithm's stability, we add Gaussian white noise with mean 0 and variance 0.5 °C to the virtual eddy.

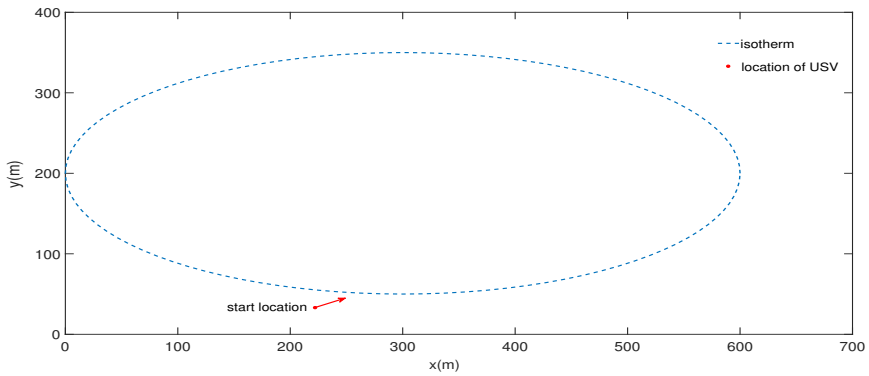


Figure 11. Fitting the isotherm of an ellipse.

Finally, we saved the relevant data regarding the turn reward changes. The reward values and step sizes under different training rounds are shown in Figures 12 and 13, respectively.

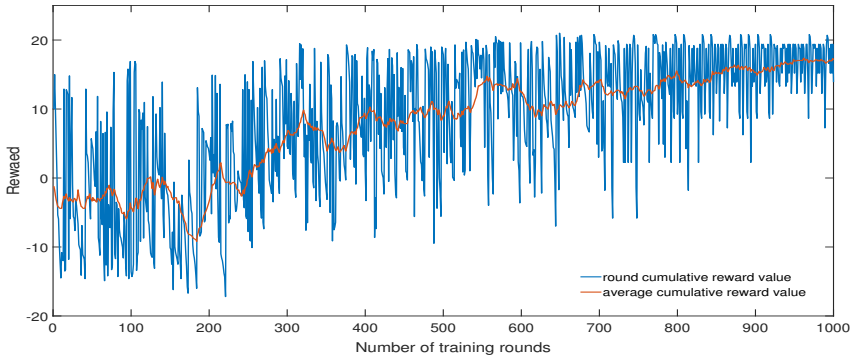


Figure 12. Variation curve of cumulative reward values and average cumulative reward values in rounds.

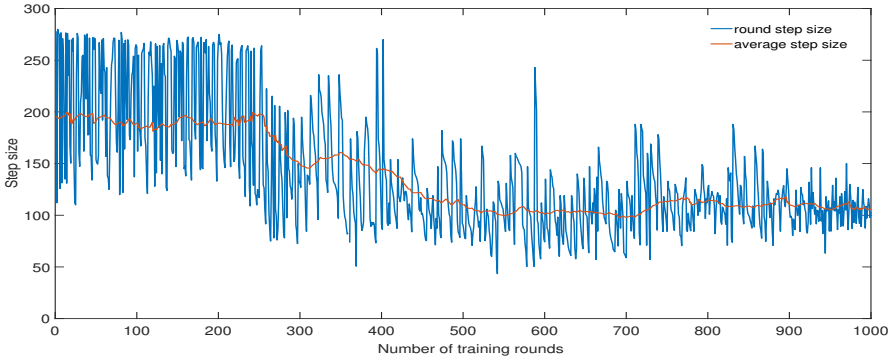


Figure 13. Turn step size and average step size variation curve.

The x and y axes in Figure 12 represent the number of training rounds and the corresponding reward values, respectively. We expected that with the increase of the number of training rounds, the cumulative total reward value and the average cumulative reward value would increase. From the curve change trend of the cumulative reward value and the average cumulative reward value (solved by the window average method of 50 rounds) in Figure 12, it can be seen that the cumulative reward value in the first 200 rounds was lower. This was because, at the beginning of the experiment, the policy network mainly output the angular velocity action of the USV according to the initialization parameter model. The algorithm was still in the stage of exploring the environment, collecting the environment sample data, and saving it to the experience playback pool. Because the output action a_t of the main network was random under the influence of noise N_t, the reward value of some rounds in the environmental exploration stage was relatively high, even close to 20. Thus, it was proved that there was a better policy in the exploration stage; that is, after the angular velocity output of part of the round, the USV was close to the isotherm. As the number of training rounds increased to 200–500, the reward value curve increased gradually. This was because, with the increase of the number of samples in the experience playback pool and the parameter iteration of the policy evaluation network, the relationship between the reward function and the action strategy was gradually established. Under the stimulation of the reward function, the strategy was also improved iteratively. The USV gradually changed from exploring environmental knowledge to making use of environmental knowledge, the algorithm converged rapidly, and the USV gradually sailed close to the isotherm.

The x and y axes in Figure 13 represent the number of training rounds and the steps for the USV to reach the isotherm, respectively. We expected that as the number of training

rounds increased, the step size would decrease. As can be seen from Figures 12 and 13, in rounds 500∼1000, USVs could constantly adjust their best course according to the environmental status information to approach the isotherm with the smallest step size. The final average cumulative reward value was stable at 15, and the average step size (solved by window average method) was also stable at 105. From the changing trend of the cumulative reward curve and step size curve, the algorithm met the task requirements of the USV tracking isotherm.

We set the sampling times to 100 and the sampling time to 1 s. As shown in Figures 14 and 15, with the increase of the number of training rounds, regardless of whether there was a bounded noise disturbance or not, the position of the USV moved closer to the isotherm. Thus, it can be seen that the algorithm showed good stability under bounded noise disturbance. The error between the actual track and the target track of the USV under different rounds without noise was selected. The track error values corresponding to different training rounds are shown in Table 3. It can be seen from Table 3 that, with the increase of training times, the output error reduced, which further proved the stability of the USV tracking isotherm control system based on the DDPG algorithm.

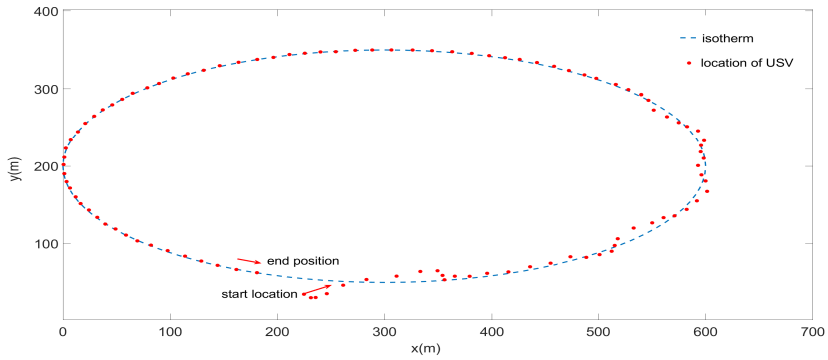


Figure 14. The location points of the USV without disturbance.

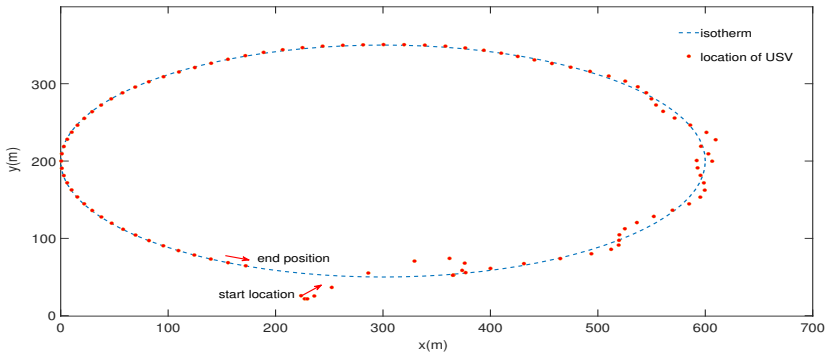


Figure 15. The location points of the USV under disturbance.

Table 3. Error table between actual track and target track of the USV.

Parameters	110	210	410	610	710	910
Average value	2.142	1.786	1.167	0.815	0.534	0.399
Variance $\left(10^{-3}\right)$	9.119	6.981	4.046	2.473	1.583	1.042

5.2. Experiment of Cooperative Control Algorithm

The input parameters included the motion state information of all current USVs and the distance information of the UAV relative to all USVs. The output parameters were the linear speed of all USVs and the UAV. The reward function included all rewards

for the confluence of USVs and the UAV, all rewards for a USV being close to the UAV, and penalties for UAV energy loss. Algorithm parameters were set as shown in Table 4.

Table 4. Parameter setting of MADDPG algorithm.

Parameters	Value
Number of training rounds max-el	20,000
Learning rate of the policy network l_{r_a}	0.001
Learning rate of the evaluation network l_{r_c}	0.0001
Discount factor γ	0.99
Capacity of experience pool R-size	1,000,000
Update parameters of the target network δ	0.01
Coefficient of reward function λ_1, λ_2	0.5, 0.5

Then, the USVs' initial coordinates were (10,0), (20,15), and (50,25), respectively. The UAV's initial coordinates were (30,30). USV_1, USV_2, and USV_3 were respectively controlled by three independent processes. When USV_3, located on the outermost isotherm, first completed the rendezvous task with the UAV, the process stopped and waited for USV_1 and USV_2 to continue to perform the confluence task. At the same time, in order to further approach the marine environment, comparative experiments of multi-agent cooperation between undisturbed and perturbed multi-agents were carried out based on the MADDPG algorithm.

Finally, we saved the relevant data regarding the turn reward changes. After 20,000 rounds of training, the experimental results without disturbance are shown in Figures 16–18. The x and y axes in Figure 16 represent the number of training rounds, reward value, and negative Q value, respectively. We expected that with the increase of the number of training rounds, the reward value and negative Q value curve would show an upward trend. As can be seen from Figure 16, the total reward value curve of the MADDPG algorithm fluctuated greatly, and the reward value was low in the first 10,000 rounds. With the increase of the number of training rounds, the relationship between the reward function and action policy was gradually established. After 10,000 rounds, the algorithm began to converge gradually, and the reward curve began to rise. It tended to stabilize after 15,000 rounds, and the final average reward value was stable at -5. At the same time, from the negative Q value (loss function) change curve of Figure 17, it can be seen that after a period of training, the negative Q value showed a downward trend. There were some fluctuations, but with the increase of the number of rounds, the Q value after 10,000 rounds gradually increased to about 0.6 and remained stable. In addition, the sampling time was set to 50 s, with sampling performed once every 1 s, and the timing reward value curve for a single round (20,000 rounds) is shown in Figure 18. The time series reward value represents the network loss function value. From the chart, we can see that the curve of the time series reward value showed a rising state and approached 0. The above three graphs prove that the training of the algorithm was stable, and the model was able to converge to the optimal situation under the current reward function setting.

As with the experiment on the USV tracking isotherm algorithm, we added Gaussian white noise with mean 0 and variance 0.5 °C to the virtual eddy; the experimental results are shown in Figures 19–21. From these three figures, we can see that the algorithm converged after adding disturbance, but the effects of the change of reward value, negative Q value, and time series reward value were worse than without disturbance. Thus, it can be seen that the perturbation of part of the agent model and the disturbance factors of the environment had an uncertain impact on the cooperative control system.

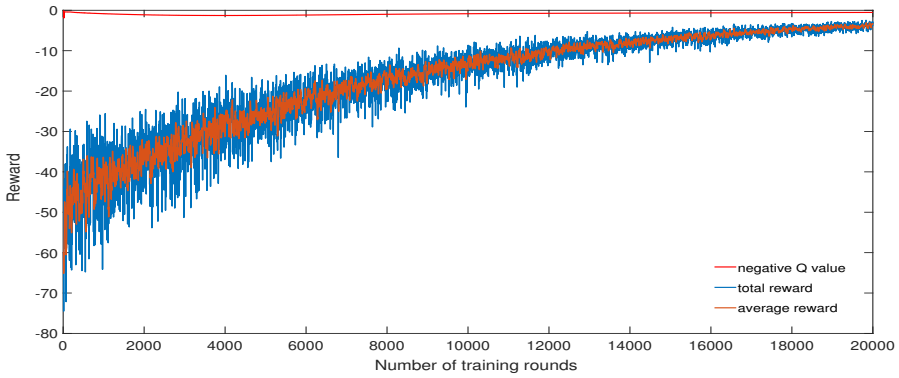


Figure 16. The change curve of the reward value after 20,000 rounds of training.

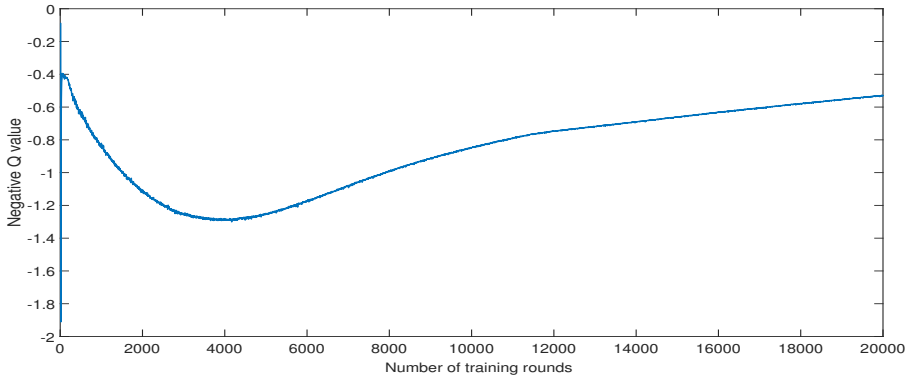


Figure 17. Negative Q value variation curve.

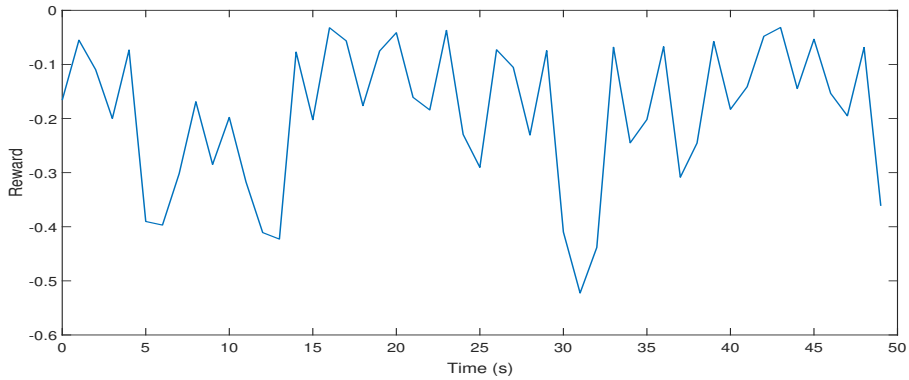


Figure 18. Timing reward for the 20,000 rounds.

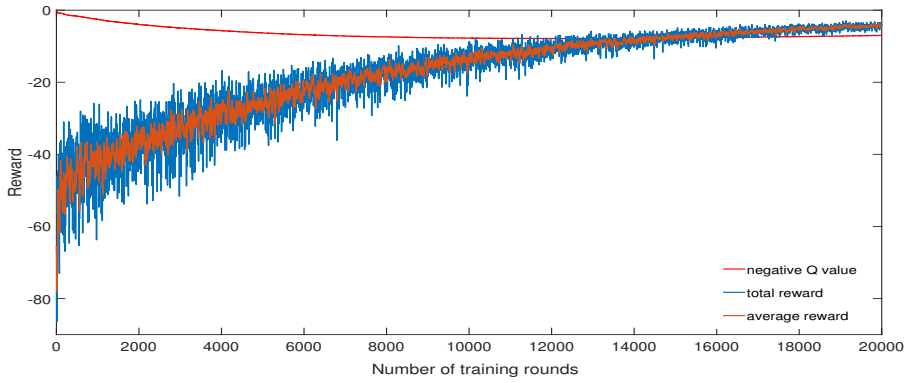


Figure 19. The variation curve of reward value in the presence of disturbance.

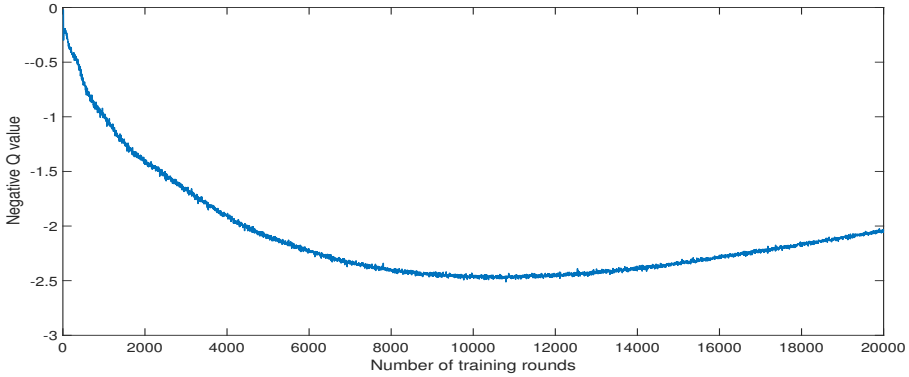


Figure 20. The variation curve of negative Q value in the presence of disturbance.

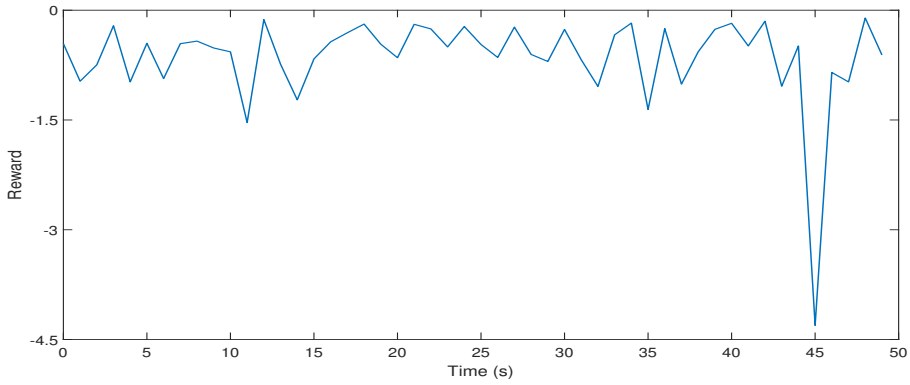
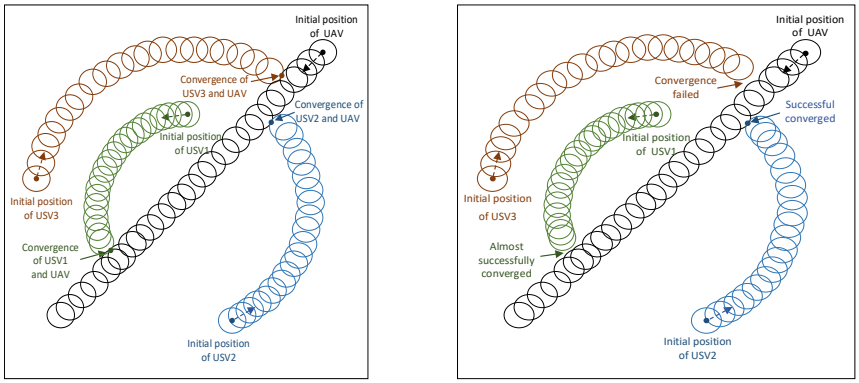


Figure 21. Time series reward under disturbance.

The two testing processes are shown in Figure 22, Figure 22a is test without noise, Figure 22b is test under noise. The brown, blue, green, and black circles in the figure represent USV_1, USV_2, USV_3, and the UAV, respectively. The brown, blue, and green dots represent the confluence of USVs and the UAV. All USVs moved along the curved isoline of water temperature, and the UAV moved in a straight line. The positions of all agents were collected every second. As shown in Figure 22, the final policy model without disturbance enabled the UAV to converge with all USVs successfully. In the last policy model test under disturbance, the convergence of USV_3 and the UAV was not successful; however, the convergence of USV_2 and the UAV was successful. USV_3 did not successfully converge with the drone, but it was close to the rendezvous point. From the test results, it can be seen that the control policy based on the MADDPG algorithm could effectively solve the problem of heterogeneous cooperative control in sea–air multi-agent observation tasks by designing appropriate reward functions. At the same time, through the comparative experimental analysis with or without disturbance, it was found that the perturbation of part of the agent model and the disturbance factors of the environment reduced the stability of the cooperative control system. Therefore, it is necessary to study the multi-agent reinforcement learning control algorithm further based on a completely model-free approach in the future.



(**a**) (**b**)

Figure 22. Test of multi-agent coordination effect with or without disturbance. (**a**) is test without noise and (**b**) is test under noise.

In order to better show our experimental process, we divided the experimental process (no Gaussian noise) into 21 pictures, as shown in Appendix A.

From the test of the multi-agent coordination effect with disturbance, we can see that we are still facing many problems, and the system may produce non-convergence under noise. We propose that this may be due to the non-static nature of the environment, the partial observability of the agent to the environment, or the parameter configuration of the algorithm itself. In addition, we plan to study the problem of the algorithm itself in the next work and combine the noise parameters in the model with the actual sea situation through a large number of actual field tests to study and explore the capability boundary of the algorithm in actual implementation. We also want to plan and study the control problem of agents in units of 100 to realize tasks with a larger area and more complex content.

6. Conclusions

Starting from the task of mesoscale eddy observation in oceanography, this paper selects and designs an algorithm for tracking isotherms with UAVs and an algorithm for reading data under the cooperative control of a UAV and USVs. Experiments show the feasibility of these algorithms, and this function and simulation system has been developed into our control system. However, before the actual operation, we are still facing the problem that the system may produce non-convergence under noise. For this reason, we will study the relationship between actual sea conditions and noise parameters and study and compare the effects of more algorithms when dealing with more complex tasks.

Author Contributions: Conceptualization, K.H., X.C., Q.X., J.J. and L.W.; methodology, K.H., X.C., Q.X., J.J. and L.W.; software, K.H., X.C., Q.X., J.J. and L.W.; validation, K.H., X.C., Q.X., J.J. and L.W.; formal analysis, K.H., X.C., Q.X., J.J. and L.W.; investigation, K.H., X.C., Q.X., J.J. and L.W.; resources, K.H., X.C., Q.X., J.J. and L.W.; data curation, K.H., X.C., Q.X., J.J. and L.W.; writing—original draft preparation, K.H., X.C., Q.X., J.J. and L.W.; writing—review and editing, K.H., X.C., Q.X., J.J. and L.W.; visualization, K.H., X.C., Q.X., J.J. and L.W.; supervision, K.H., X.C., Q.X., J.J. and L.W.; project administration, K.H., X.C., Q.X., J.J. and L.W.; funding acquisition, K.H., X.C., Q.X., J.J. and L.W.; All authors have read and agreed to the published version of the manuscript.

Funding: Research in this article is supported by Joint Fund of Science and Technology Department of Liaoning Province and State Key Laboratory of Robotics (No. 2021-KF-22-19) China; the key special project of the National Key *R&D* Program (2018YFC1405703), for which I would like to express my heartfelt thanks.

Institutional Review Board Statement: Ethical review and approval were waived for this study due to the data being provided publicly.

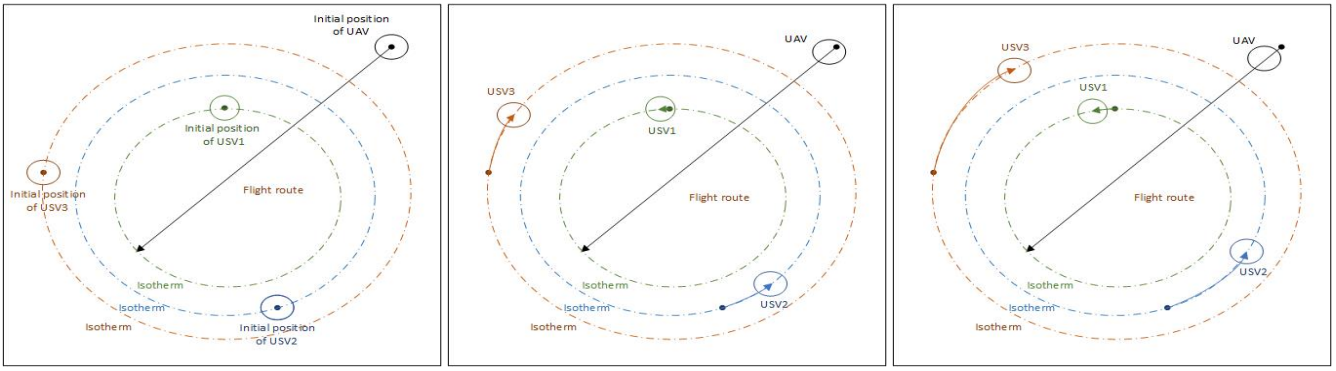
Informed Consent Statement: Not applicable.

Data Availability Statement: The data and code used to support the findings of this study are available from the corresponding author upon request (fenix@jlxy.nju.edu.cn).

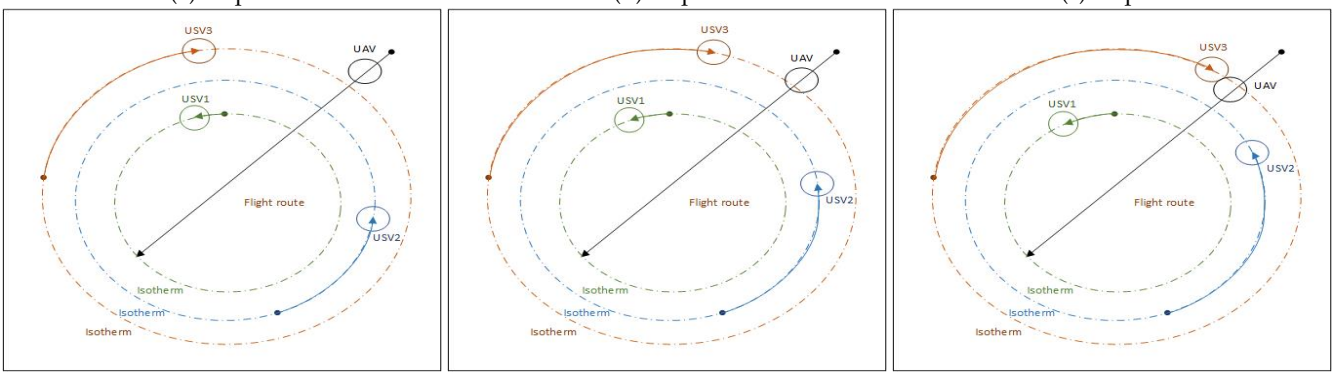
Acknowledgments: I would like to express my heartfelt thanks to the reviewers who gave valuable revisions to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

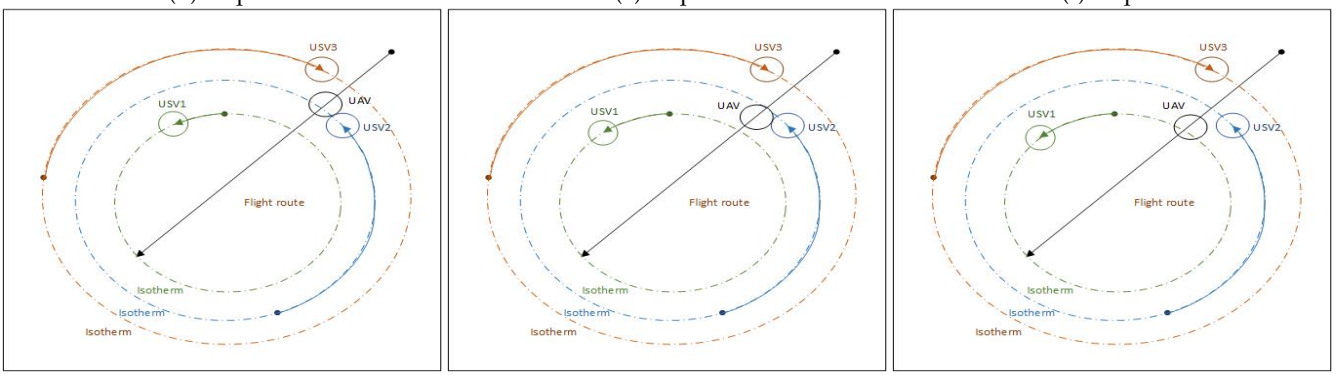
Appendix A



(**a**) Step 1. (**b**) Step 2. (**c**) Step 3.

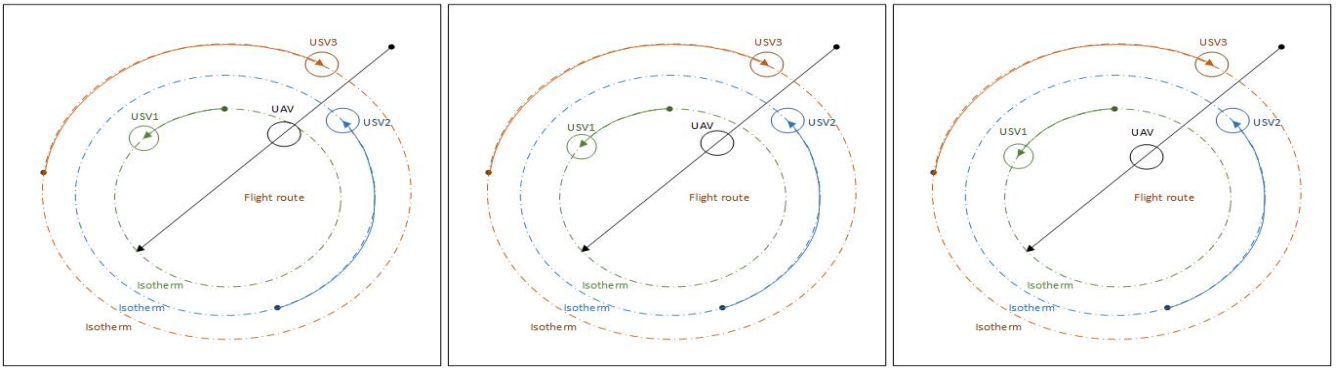


(**d**) Step 4. (**e**) Step 5. (**f**) Step 6.



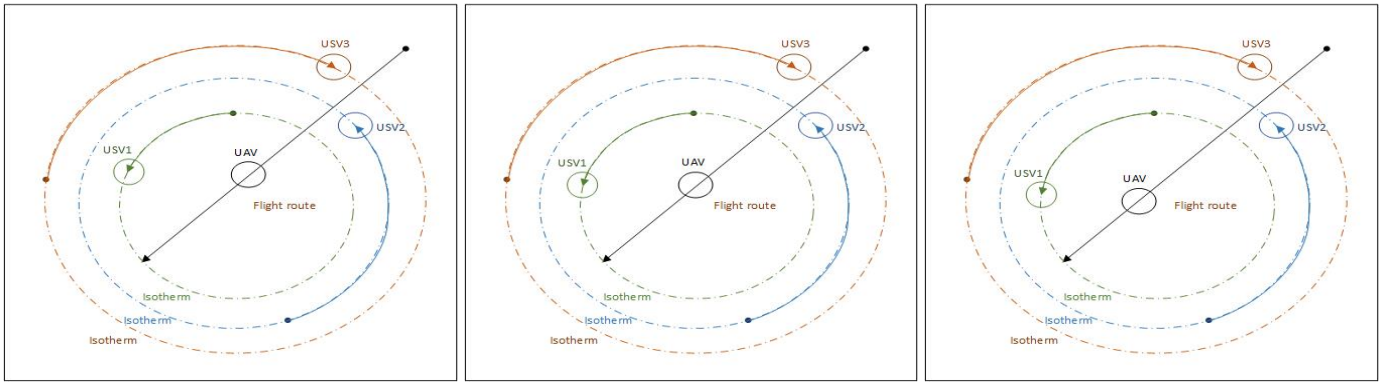
(**g**) Step 7. (**h**) Step 8. (**i**) Step 9.

Figure A1. *Cont.*

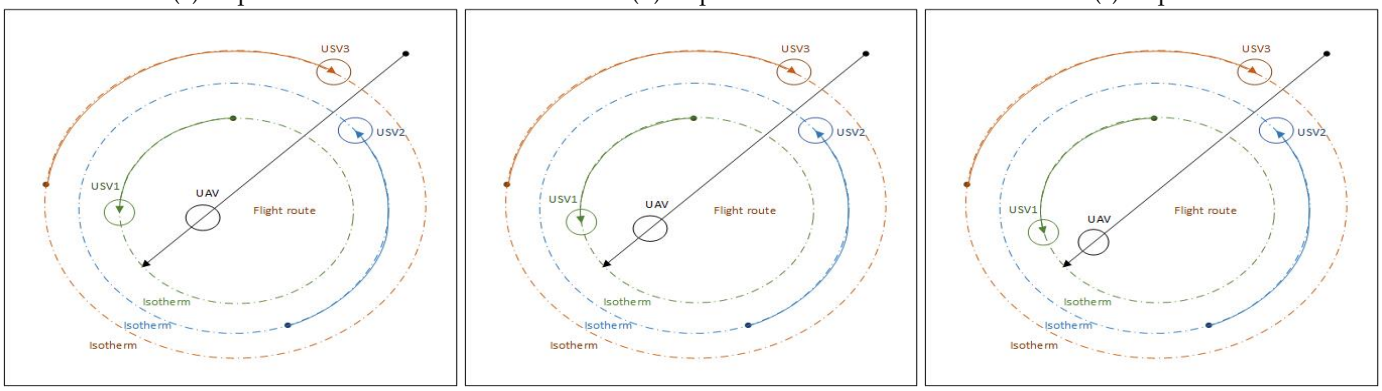


(**j**) Step 10. (**k**) Step 11. (**l**) Step 12.

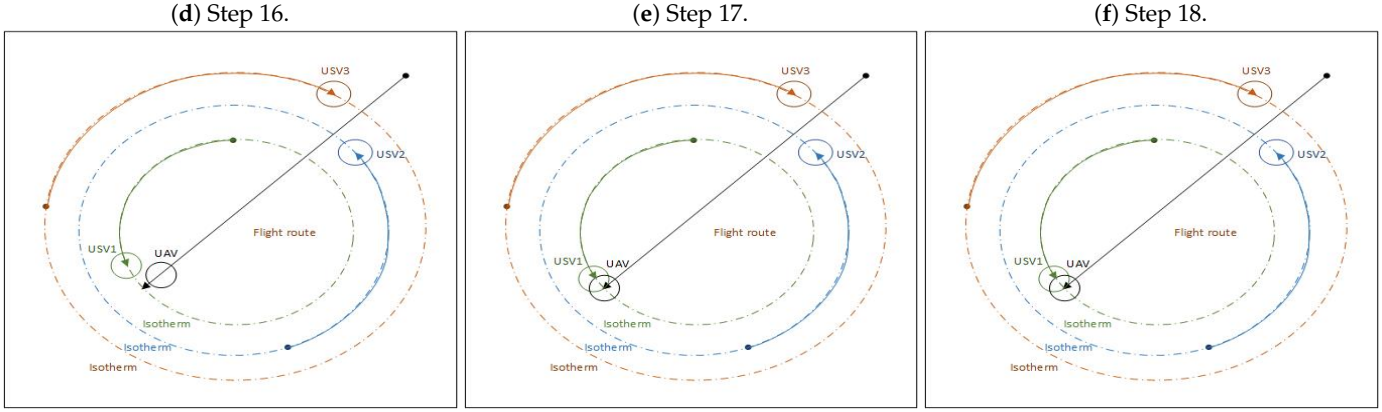
Figure A1. Experiment of cooperative control algorithm process without noise (Step 1–Step 12).



(**a**) Step 13. (**b**) Step 14. (**c**) Step 15.



(**d**) Step 16. (**e**) Step 17. (**f**) Step 18.



(**g**) Step 19. (**h**) Step 20. (**i**) Step 21.

Figure A2. Experiment of cooperative control algorithm process without noise (Step 13–Step 21).

References

1. Burk, S.D.; Thompson, W.T. Mesoscale eddy formation and shock features associated with a coastally trapped disturbance. In Proceedings of the Oceans 2003. Celebrating the Past. . . Teaming Toward the Future (IEEE Cat. No. 03CH37492), San Diego, CA, USA, 22–26 September 2003; pp. 1763–1770.
2. Vu, M.T.; Van, M.; Bui, D.H.P.; Do, Q.T.; Huynh, T.-T.; Lee, S.-D.; Choi, H.-S. Study on Dynamic Behavior of Unmanned Surface Vehicle-Linked Unmanned Underwater Vehicle System for Underwater Exploration. *Sensors* **2020**, *20*, 1329. [CrossRef] [PubMed]
3. Cho, H.; Jeong, S.-K.; Ji, D.-H.; Tran, N.-H.; Vu, M.T.; Choi, H.-S. Study on Control System of Integrated Unmanned Surface Vehicle and Underwater Vehicle. *Sensors* **2020**, *20*, 2633. [CrossRef] [PubMed]
4. Jung, D.W.; Hong, S.M.; heon Lee, J.; Cho, H.J.; Choi, H.S.; Vu, M.T. A Study on Unmanned Surface Vehicle Combined with Remotely Operated Vehicle System. *Proc. Eng. Technol. Innov.* **2018**, *9*, 17–24.
5. Polvara, R.; Sharma, S.; Wan, J.; Manning, A.; Sutton, R. Autonomous vehicular landings on the deck of an unmanned surface vehicle using deep reinforcement learning. *Robotica* **2019**, *37*, 1867–1882. [CrossRef]
6. Liu, Y.-J.; Cheng, S.-M.; Hsueh, Y.-L. eNB selection for machine type communications using reinforcement learning based Markov decision process. *IEEE Trans. Veh. Technol.* **2017**, *66*, 11330–11338. [CrossRef]
7. Lin, S.-W.; Huang, Y.-L.; Hsieh, W.-K. Solving Maze Problem with Reinforcement Learning by a Mobile Robot. In Proceedings of the 2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE), Fujian, China, 8–10 November 2019; pp. 215–217.
8. Woo, J.; Yu, C.; Kim, N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean Eng.* **2019**, *183*, 155–166. [CrossRef]
9. Bouhamed, O.; Ghazzai, H.; Besbes, H.; Massoud, Y. Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
10. Wu, X.; Liu, S.; Zhang, T.; Yang, L.; Li, Y.; Wang, T. Motion control for biped robot via DDPG-based deep reinforcement learning. In Proceedings of the 2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA), Beijing, China, 15–19 August 2018; pp. 40–45.
11. Werbos, P.J.; Miller, W.T.; Sutton, R.S. A menu of designs for reinforcement learning over time. *Neural Netw. Control* **1990**, *3*, 67–95.
12. Modares, H.; Lewis, F.L. Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* **2014**, *50*, 1780–1792. [CrossRef]
13. Passalis, N.; Tefas, A. Deep reinforcement learning for controlling frontal person close-up shooting. *Neurocomputing* **2019**, *335*, 37–47. [CrossRef]
14. Xia, M.; Wang, T.; Zhang, Y.; Liu, J.; Xu, Y. Cloud/shadow segmentation based on global attention feature fusion residual network for remote sensing imagery. *Int. J. Remote Sens.* **2021**, *42*, 2022–2045. [CrossRef]
15. Xia, M.; Wang, K.; Song, W.; Chen, C.; Li, Y. Non-intrusive load disaggregation based on composite deep long short-term memory network. *Expert Syst. Appl.* **2020**, *160*, 113669. [CrossRef]
16. Chen, B.; Xia, M.; Huang, J. Mfanet: A multi-level feature aggregation network for semantic segmentation of land cover. *Remote Sens.* **2021**, *13*, 731. [CrossRef]
17. Jiang, X.; Ji, Y. HD3: Distributed Dueling DQN with Discrete-Continuous Hybrid Action Spaces for Live Video Streaming. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 2632–2636.
18. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International conference on machine learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
19. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **2000**, *12*, 1057–1063.
20. Bhatnagar, S. An actor–critic algorithm with function approximation for discounted cost constrained Markov decision processes. *Syst. Control Lett.* **2010**, *59*, 760–766. [CrossRef]
21. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International conference on machine learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
22. Qi, C.; Hua, Y.; Li, R.; Zhao, Z.; Zhang, H. Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing. *IEEE Commun. Lett.* **2019**, *23*, 1337–1341. [CrossRef]
23. Mnih, V. , Kavukcuoglu, K. , Silver, D. , Graves, A. , Antonoglou, I. , Wierstra, D. , Riedmiller, M . Playing Atari with Deep Reinforcement Learning. Computer Science. *arXiv* **2013** arXiv:1312.5602.
24. Zhou, X.; Wu, P.; Zhang, H.; Guo, W.; Liu, Y. Learn to navigate: Cooperative path planning for unmanned surface vehicles using deep reinforcement learning. *IEEE Access* **2019**, *7*, 165262–165278. [CrossRef]
25. Lillicrap, T. P. ; Hunt, J. J. ; Pritzel, A. ; Heess, N. ; Erez, T. ; Tassa, Y. ;Silver,D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
26. Wang, D.; Shen, Y.; Sha, Q.; Li, G.; Kong, X.; Chen, G.; He, B. Adaptive DDPG design-based sliding-mode control for autonomous underwater vehicles at different speeds. In Proceedings of the 2019 IEEE Underwater Technology (UT), Kaohsiung, Taiwan, 16–19 April 2019; pp. 1–5.

27. Xing, S.; Guan, X.; Luo, X. Trajectory tracking and optimal obstacle avoidance of mobile agent based on data-driven control. In Proceedings of the 29th Chinese Control Conference, Beijing, China, 29–30 July 2010; pp. 4619–4623.
28. Zhang, H.; Jiang, H.; Luo, Y.; Xiao, G. Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. *IEEE Trans. Ind. Electron.* **2016**, *64*, 4091–4100. [CrossRef]
29. Foerster, J.N.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. *arXiv* **2017**, arXiv:1705.08926.
30. Xiong, S.; Hou, Z.; Yu, X. Data-driven Formation Control for a Class of Unknown Heterogeneous Discrete-time MIMO Multi-agent System with Switching Topology. In Proceedings of the 2019 12th Asian Control Conference (ASCC), Fukuoka, Japan, 9–12 June 2019; pp. 277–282.