sensors

MDPI

Article

Dynamic Navigation and Area Assignment of Multiple USVs Based on Multi-Agent Deep Reinforcement Learning

Jiayi Wen, Shaoman Liu * and Yejin Lin

Lab of Intelligent Marine Vehicles of DMU, Dalian Maritime University, Dalian 116026, China
* Correspondence: lsm1015@dlmu.edu.cn

Abstract: The unmanned surface vehicle (USV) has attracted more and more attention because of its basic ability to perform complex maritime tasks autonomously in constrained environments. However, the level of autonomy of one single USV is still limited, especially when deployed in a dynamic environment to perform multiple tasks simultaneously. Thus, a multi-USV cooperative approach can be adopted to obtain the desired success rate in the presence of multi-mission objectives. In this paper, we propose a cooperative navigating approach by enabling multiple USVs to automatically avoid dynamic obstacles and allocate target areas. To be specific, we propose a multi-agent deep reinforcement learning (MADRL) approach, i.e., a multi-agent deep deterministic policy gradient (MADDPG), to maximize the autonomy level by jointly optimizing the trajectory of USVs, as well as obstacle avoidance and coordination, which is a complex optimization problem usually solved separately. In contrast to other works, we combined dynamic navigation and area assignment to design a task management system based on the MADDPG learning framework. Finally, the experiments were carried out on the Gym platform to verify the effectiveness of the proposed method.

Keywords: USV; trajectory design; policy gradient; multi-agent deep reinforcement learning; multi-object optimization

Citation: Wen, J.; Liu, S.; Lin, Y. Dynamic Navigation and Area Assignment of Multiple USVs Based on Multi-Agent Deep Reinforcement Learning. *Sensors* **2022**, *22*, 6942. https://doi.org/10.3390/s22186942

Academic Editor: Kristijan Lenac

Received: 18 July 2022
Accepted: 7 September 2022
Published: 14 September 2022

1. Introduction

Nowadays, various kinds of unmanned robots are developing rapidly with the arrival of the 5G era. Typical robots include the autonomous underwater vehicle (AUV), unmanned surface vessel (USV), unmanned ground vehicle (UGV), and unmanned aerial vehicle (UAV) [1–4], which play important roles in the artificial intelligence (AI)-enabled next-generation (6G) network. As an important unmanned marine tool, the USV is also a promising technique to provide wireless communication due to its low cost and high flexibility [5,6]. Especially when it comes to complex tasks, such as maritime joint search and rescue, maritime multi-target search, the construction of marine information networks, and other missions. However, it is difficult to rely on one single USV to complete the task, while multi-USV cooperation can be a good solution to the problem. This paper studies the multi-USV dynamic path planning and area assignment problem, which will effectively improve the autonomy and reliability of the USV in the increasing field.

As a generic technology among different robotics systems, the optimization objectives mainly include path length, time or energy consumption, risk measure, maneuverability, etc. They are classified into four kinds based on the model of consumption space, i.e., the grid-based methods [7–10] solved by the heuristic strategies, such as the A* algorithm, D* algorithm, Dijkstra algorithm, and Q-learning algorithm in reinforcement learning theory, or intelligent strategies represented by particle swarm optimization and evolutionary algorithms; the sampling-based methods such as probabilistic roadmap, rapidly-exploring random tree (RRT) [11–13], or deep reinforcement learning method represented by experience replay based DQN algorithm; the mathematical optimization methods such as mixed integer linear programming (MILP) and model predictive control (MPC) [14–16]; and the

potential field methods such as the artificial potential field (APF) and the interfered fluid dynamical system (IFDS) [17–19].

In addition to the classification mentioned above, when the purpose of agents is taken as the dividing boundaries, the path planning problem is divided into the single agent path planning problem (SAPP) and the multi-agent path planning problem (MAPP). For the requirement of accessing a series of intermediate points during the monitoring mission in SAPP, Ning Wang et al. [20] proposed a successive waypoints tracking method using a BT-guided model-free solution to solve the SWT problem. Yong et al. [21] proposed the dynamic augmented multi-objective particle swarm optimization algorithm to solve the problem that obstacles and water flows exist at the same time to find the shortest and safest route to the target, which is subject to collision avoidance. The rapidly random-exploring tree (RRT) algorithm and its variants are some of the more popular path planning methods in SAPP. However, they suffer sensitivity to the initial solution, which requires a lot of memory and time to converge to the optimal solution. In order to solve this problem, Wang et al. [22] propose the NRRT* to achieve nonuniform sampling in the path planning process by learning quantities of successful planning cases from the A* algorithm. Thus, the sampling process is guided, and the efficiency of the algorithm is improved. The sampling mechanism is also used in reinforcement learning which has been prevalent in recent years. In [23], Tom Schaul et al. proposed the prioritized experience replay method to speed up the convergence of the training process, and the effectiveness was verified on the Gym platform.

For the multi-objective optimization problem in SAPP, Hongqiang Sang et al. [24] proposed a novel deterministic algorithm named the multiple sub-target artificial potential field (MTAPF) based on the heuristic A* algorithm. The optimal path is divided by this algorithm into multiple sub-target points to form a sub-target point sequence. Ning Wang et al. [25] proposed a multilayer path planner (MPP) with global path-planning (GPP), collision avoidance (CA), and routine correction (RC) for an unmanned surface vehicle (USV). In addition, some methods inspired by these problems or traditional search strategies are suggested, such as the bridge access path-planning method [26,27].

However, it is difficult to complete tasks only by relying on one single agent in certain scenarios, such as maritime joint search and rescue, maritime multi-target search, the construction of the marine information network, etc. Thus, the MAPP approach becomes another effective method to deal with complex multi-objective problems. Yu Wu et al. [28] proposed a new cooperative path planning algorithm based on an improved particle swarm optimization (IPSO) algorithm aimed at maximizing the search space, minimizing the terminal error, and generating paths in a centralized or distributed mode. In the same way, Pradhan B. et al. [29] realized multi-robot navigation tasks by combining particle swarm optimization (PSO) with the feed forward neural network (FFNN). To optimize search capability, Yu Wu et al. [30] proposed a clustering improved ant colony optimization (CIACO) algorithm, which strengthens the global and local search ability in the early and later phases of iterations. Xinghai Guo et al. [31] proposed a chaotic and sharing-learning particle swarm optimization (CSPSO) algorithm. The path planning problem is divided into two stages: global path planning and path control, to solve the extended TSP, and the nonlinear multi-objective model. For the multi-objective joint optimization problem, Milad Nazarahari et al. [32] proposed an enhanced genetic algorithm (EGA) to improve the initial paths in continuous space and jointly optimize the path length, smoothness, and safety of the agent.

However, the alternating algorithm may not converge as the number of optimized variables of USV is increased [33]. In addition, the optimized results can only be used for the current environment, while, when the environment changes, the proposed optimization algorithms can become invalid.

Deep reinforcement learning (DRL), as a branch of artificial intelligence (AI), provides an alternative solution for such complex optimization problems, such as resource allocation for V2V communications [34], the stochastic shortest path (SSP) problem [35], and mode

selection and resource allocation for the fog radio access networks [36]. The DRL method can deal with a large state space and time-varying environments [37].

However, in the multi-agent reinforcement learning scenario, each agent is unstable and the environment is changing. In this paper, we propose to use the multi-agent deep deterministic policy gradient (MADDPG) algorithm to solve the area assignment and dynamic trajectory design problem in a multi-USV cooperation task. The algorithm utilized centralized training within the decentralized execution framework [38]. Each USV can be regarded as an agent. The contributions of this paper can be summarized as follows:

- We propose a multi-agent DRL method for the joint optimization problem in the multi-USV cooperation scenario, where they share the common reward function to achieve the maximum success rate of the system. The number of UAVs can be arbitrary in the proposed MADRL algorithm, while the conventional methods can only deal with the simple case, i.e., no more than two USVs;
- We consider a scenario in which obstacles change position at fixed intervals and the USV needs to adjust its actions in real-time to avoid collisions with obstacles and other USVs, ultimately achieving the task of dynamic trajectory design and area assignment. Our algorithm is designed for 2D space, in which the trajectory of the UAV can be shown in 2D;
- We develop an efficient task management framework, which adopts centralized training and the decentralized execution method. In order to improve the learning ability of the agent and better adapt to the environment, we add the "soft update" mechanism in MADDPG, so that the target network can better track the learning policy. All USVs work in a cooperative way to achieve the reasonable allocation of target areas. In order to maximize cooperative rewards, all experiences are trained together, while all behaviors are at the disposal of the USV itself. The output of the actor network is the action of the USV, which is based on the USV's own observation.

The rest of the paper is organized as follows: In Section 2, we present the system model and problem formulation. Section 3 demonstrates the MADDPG method for the cooperative multi-USV network. Section 4 shows the simulation results and discussions. Finally, Section 5 elaborates on the conclusion of this paper.

2. System Model

This paper considered a multi-USV cooperative dynamic navigation and area allocation scenario with random obstacles on the sea surface, K USVs, N obstacles, and K target areas, as shown in Figure 1. Each USV departs from the same point and adjusts the target area to be reached according to the safety of the surrounding environment and the distance required to reach the target area. Specifically, each USV should keep a certain distance from other USVs during navigation to ensure communication as well as safety. At the same time, when encountering obstacles, it can smoothly avoid obstacles under the premise of keeping the maximum communication distance with other USVs. In order to simulate the real dynamic port situation, obstacles involved dynamic obstacles and static obstacles, among which dynamic obstacles would change their positions randomly after each training episode.

Next, we will describe the multi-USV system and the problem formulation for the proposed multi-USV cooperative dynamic navigation and area allocation scenario.

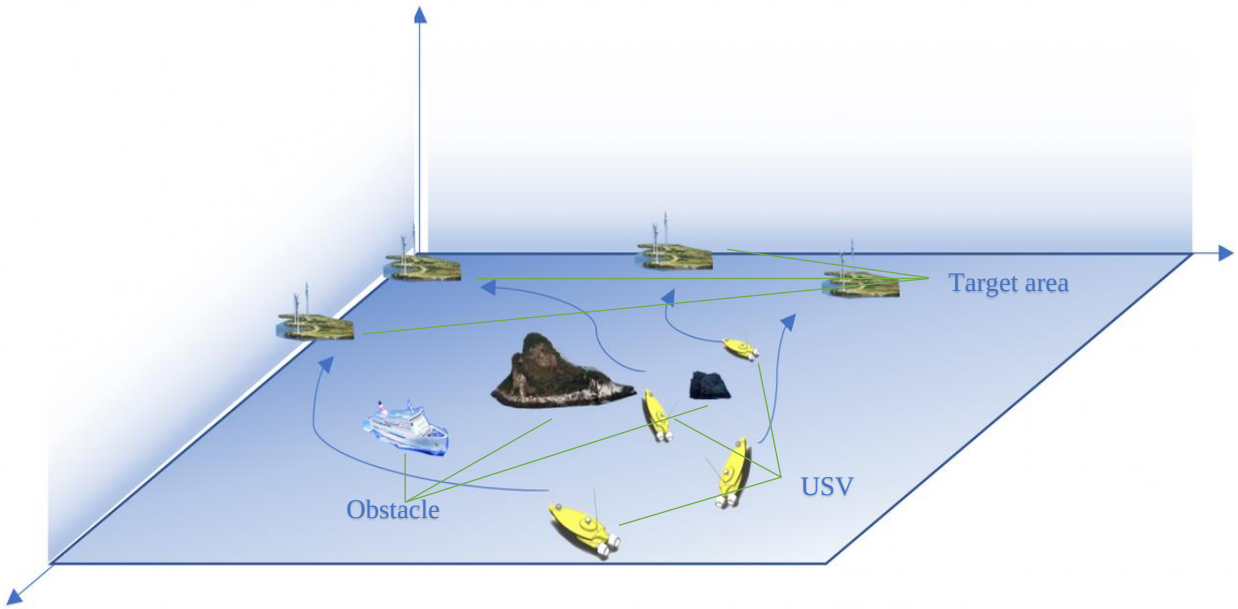

Figure 1. Multi-USV communication system.

2.1. Multi-USV System

We considered a two-dimensional environment in which K USVs were represented by U, each USV's position at time t is denoted by $\left(x_i^U(t), y_i^U(t)\right)$, and the matching target is represented by T_i, among which the obstacle is denoted by O_i, and the positions of targets and obstacles are represented by $\left(x_i^T, y_i^T\right)$ and $\left(x_i^O, y_i^O\right)$, respectively. For an arbitrary USV, the path taken can be denoted by $P_i = \left\{\left(x_i^U(0), y_i^U(0)\right), \left(x_i^U(1), y_i^U(1)\right), \ldots, \left(x_i^U(n), y_i^U(n)\right)\right\}$ and the path length by $\sum_{i}^{K_U} d_i$. Thus, the discrete dynamics model of each USV can be expressed as:

$$\begin{cases} v_i^{t+1} = v_i^t + \frac{F_i^t}{m}\Delta t \\ p_i^{t+1} = p_i^t + v_i^t \Delta t \end{cases} \tag{1}$$

where v_i^t and p_i^t, respectively, represent the speed and position of the i_{th} USV at time t, and Δt represents the sampling period where F is a force vector. The distance between the USV and the obstacle is represented by d_i^O, and the distance between USVs is represented by d_i^U, while the distance from the USV to the matching target is represented by d_i^T.

2.2. Problem Formulation

In the given system model, our goal is to enable all USVs to autonomously navigate to their respective target areas by optimizing all trajectories, and each target is only assigned to one USV. Then, we have

$$\begin{cases} \bigcup_{i=1}^{K_U} T_i' = \mathbf{T}, i \in \{1, \ldots, K_T\} \\ \forall i \neq j, T_i' \neq T_j' \; i, j \in \{1, \ldots, K_T\} \end{cases} \tag{2}$$

where $\mathbf{T}$ represent all assignable regions. All USVs can navigate by adjusting their positions dynamically to avoid static and dynamic obstacles, which can be expressed as

$$\forall i, j, O_j \notin P_i \; i \in \{1, \ldots, N\}, j \in \{1, \ldots, U_K\} \tag{3}$$

During the mission, each USV needs to maintain a certain distance to stay within communication distance. We utilized Z to describe the communication graph of K USVs, where E is the edge set, defined as

$$E = \left\{ (U_i, U_j) \middle| d(r_i, r_j) = p_i - p_j \leq \rho_c \right\} \tag{4}$$

where ρ_c is the maximum communication distance. In other words, when the distance between two USVs is less than the maximum communication distance, there will be an undirected edge between them, and they can communicate with each other. Accordingly, each USV collaboratively adjusts its position and selects the target area based on the hazards of its environment. The collision-free path is then formulated as:

$$\forall i, j, \left(x_i^U(t), y_i^U(t) \right) \neq \left(x_j^U(t), y_j^U(t) \right) \tag{5}$$

Thus, in this article, a multi-agent deep reinforcement learning-based trajectory design and area assignment algorithm are proposed.

3. MADDPG Approach for Cooperative Multi-USV Network

The multi-USV collaboration optimization issue is challenging as it requires a joint kinetic USV trajectory and area assignment. In fact, this is NP-hard, and the approach of exhaustion is usually invalid for such a multiple USVs scenario. As the quantity of USVs increases, the calculation intricacy will elevate remarkably. As far as we know, there is research utilizing the traditional optimization algorithm to tackle this problem in these intricate settings. The MARL can solve this dynamic multi-objective optimization problem effectively and accurately. Therefore, in this section, we will delineate how to utilize the MARL approach to tackle the multi-USV collaboration issue. In Section 3.1, we formulate the proposed problem as a Markov process and define the agent, state, action, and reward separately. Section 3.2 presents the MADDPG algorithm for the multi-USV joint optimization scenario.

3.1. Markov Game for Multi-USV Cooperation

To study the best behavior mode in given states, we employ a Markov decision process (MDP) [39] to formulate the AUV motion planning, as the Markov decision process (MDP) offers a mathematical framework for simulating the randomized strategies that can be implemented with Markov states in a given scene. The method is modeled as tuples (S, A, R, P, γ), and in this case, the tuples are action set A, state set S, state transition probability P, discount factor γ, and reward function R of the system. The action selection of the agent, a functional mapping from each state $s_i^t \in \mathcal{S}$ to action $a_i^t \in \mathcal{A}$, is modeled as policy π. The value function $v_\pi(s)$ of the process is defined as the expected sum of discount rewards that act continuously from and along with state s. In case a policy π can attain the best from any initial state $s_i^t \in \mathcal{S}$, we detect it as an optimal policy π^*. This is the target of this training of the deep neural network with a deep reinforcement learning module.

Agent: Each USV can be considered an agent. USVs maintain a certain distance from each other during navigation to facilitate communication, and each USV obtains a state containing its own information and the information of the surrounding environment in the process of interaction with the environment. The environment in this problem is completely observed; therefore, the observations are equivalent to the state. Each actor has its own actor and critic network which act as the executor and evaluator of the policy, respectively. Each agent observes its own state, takes actions according to its own strategy, and then obtains rewards from the environment to reach the next state, as shown in Figure 2.

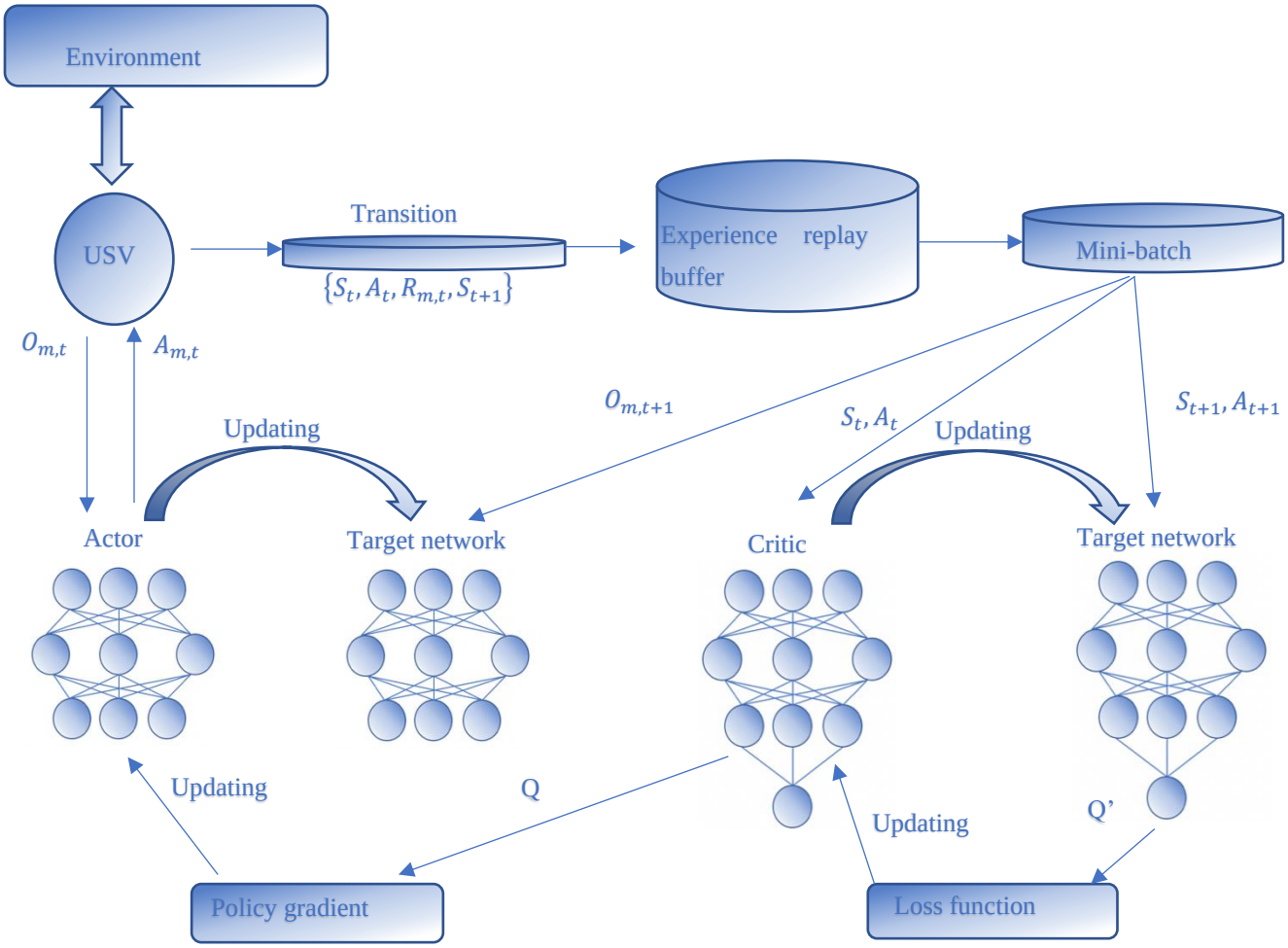

Figure 2. The framework of MADDPG for the cooperative multi-USV network.

State: The state of any agent is a one-dimensional vector with five components, $[\mathrm{x}, \mathrm{y}, d_i^U, d_i^O, d_i^T]$. The first two elements represent the current 2D position of the agent, the third entry d_i^U represents the distance between agents, and the fourth entry d_i^O represents the distance between agent i and the surrounding obstacles. The fifth element d_i^T represents the distance from agent i to the current matching target which can be changed with the changes in the USV's selection.

Action: The action for each agent is the output of its actor network, which is expressed by $v_i = u_i(o_i)$. This is a Gaussian distribution where v_i is composited by the velocities of the x and y axes. An action is executed by the formula $\vec{a} = \vec{v} * \Delta t + \mathcal{N}_t$, where $\vec{a}$ is a displacement vector and $\mathcal{N}_t$ is a random noise. The action is determined by the position of the target area and the risk factors of the surrounding environment.

Reward: Each USV obtains its own reward, which depends on the current state, current actions, and the next state (s, a, s') at each time slot t. In our proposed multi-USV scenario, the reward consists of four parts: communication distance $l^{\max}$ limitation penalty, distance penalty, threat area penalty, and collision penalty. We utilized R_l to represent the communication limitation reward, which depended on the next state of the agent. There will be a large minus penalty $-R_l$ for the agent if the agent is out of the communication space in the next state. The space limitation is set as follows

$$0 \leq X_{k,t} \leq l^{\max}, \forall k \in K, t \in T \tag{6}$$

and

$$0 \leq Y_{k,t} \leq l^{\max}, \forall k \in K, t \in T \tag{7}$$

where $X_{k,t}$ and $Y_{k,t}$ represent the horizontal and vertical coordinates of USV, respectively. In the same way, we denote $R_{k,o}$ as the collision penalty, which can be described as

$$\begin{cases} R_{k,o} = -\sum_{k=1}^{K} z_{k,o,t} P_{k,o}(t) \\ z_{k,o,t} = \{0,1\}, \forall k \in K, o \in O, t \in T \end{cases} \tag{8}$$

where $z_{k,o,t} = 1$ means there has been a collision, and $R_{k,o}(t)$ is the collision penalty of USV. Additionally, we set up a threat area penalty $R_{k,r}$ to maintain a certain communication distance between the USVs and prevent collisions. Then, we have

$$R_{k,r} = \mathrm{d}_{\min(u_i,u_j)} + \sigma - \mathrm{d}_{u_i^t,u_j^t} \tag{9}$$

where $\mathrm{d}_{\min(u_i,u_j)} =$ agent.size + agent.size, σ represents the width of critical area of agents, and $\mathrm{d}_{u_i^t,u_j^t}$ represents the distance between agents. In order to guide the USV closer to the target, we give the USVs a reward dependent on the distance between the USV and the target as

$$R_{k,t} = \sum_{k=1}^{K} \frac{k_{dis}}{\sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}} \tag{10}$$

where $R_{k,t}$ is the distance reward for USV, and $(x_i - x_t)$, $(y_i - y_t)$ are the coordinates of the USV. Moreover, k_{dis} is the hyperparameters set manually. Thus, the whole reward $R_{w,t}$ is

$$R_{w,t} = R_l + R_{k,o} + R_{k,r} \tag{11}$$

3.2. Multi-Agent DDPG Approach

In the collaborative MARL scenario, each agent engages in interactions with the environment and acquires team rewards to facilitate collaboration. Single-agent reinforcement learning algorithms, such as proximal policy optimization (PPO) and the deep Q network (DQN), can be straightly utilized to deal with collaborative scenarios by letting each agent learn its optimum Q function in an independent manner. Nevertheless, the environment can be unstable from the perspective of any agent. To tackle the unstable issue in MARL, we utilized the MADDPG algorithm [38] learning a centralized Q-function for each agent as per the global information. When all the actions of the agents are known, the environment is stable. This is due to

$$\mathcal{P}(s' \mid s, a_1, \ldots, a_L, \pi_1, \ldots, \pi_L) = \mathcal{P}(s' \mid s, a_1, \ldots, a_L) = \mathcal{P}(s' \mid s, a_1, \ldots, a_L, \pi_1', \ldots, \pi_L') \tag{12}$$

for any $\pi_i \neq \pi_i'$, in which π_i is the policy of agent i.

The aim of each agent is to select the beneficial policy that maximizes the accumulated reward $J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[R]$ prior to the description of the MADDPG arithmetic. Let us start with the policy gradient (PG) algorithm utilized for the continuous control issue. The crucial goal of the PG is to straightly adjust the parameter θ of policy π at the orientation of $\nabla_\theta J(\theta)$, which is expressed as

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) dsda = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a)] \tag{13}$$

where the policy π_θ is stochastic. At each step, the action is sampled as per the conditional probability density $\pi_\theta(a|s)$. When the action space dimension is very large, the PG algorithm may need more samples, which will induce a remarkable computational challenge. Unlike the random policy requiring the exploration of the full state and action space in (12), the deterministic policy gradient (DPG) takes a deterministic policy into consideration $\mu_\theta : S \to A$, which merely needs to integrate over the state space, as

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^{\mu}(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s,a)\Big|_{a=\mu_\theta(s)} ds = \mathbb{E}_{s\sim p^{\mu}}\Big[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s,a)|_{a=\mu_\theta}(s)\Big] \tag{14}$$

It is obvious from (14) that the DPG arithmetic prevents the integral over the entire action space, which can decrease the computational intricacy and ameliorate the training efficiency. In contrast to PG, DPG can tackle the difficult enhancement issue with high-dimensional actions.

As the extension of DPG, DDPG adopts the deep neural network (DNN) to approximate the policy μ and critic $Q^{\mu}(s,a)$. Nevertheless, the updated network $Q^{\mu}\left(s,a\middle|\theta^{Q}\right)$ cannot be straightly utilized for estimating the target value, which will induce the update Q divergent. Hence, the DDPG algorithm utilizes the soft target updates rather than straightly copying the weights of the update Q.

Specifically, DDPG utilized a copy of the actor and critic networks to calculate the target value, which is denoted as $Q^{'}(s,a)$ and $\mu_\theta^{'}(s)$, respectively. For the sake of suiting the environment and ameliorating the exploration efficiency, another trick of DDPG is to add stochastic noise into the actor policy, as presented in Algorithm 1.

Algorithm 1. Training algorithm using the MADDPG framework.

Parameters: batch size β, training episodes. M, training step T, action noise $\mathcal{N}$ and the number of USVs N, actor networks' weights for each agent $\theta_i^{\mu^{'}}$

1: **For** episode = 1 to M **do**

2: Initialize observations O_{init} , $O_{\text{new}} \leftarrow O_{\text{init}}$.
Initialize: Actor network μ, critic network Q with weights θ^{μ}, θ^{Q}.
Initialize: Target actor, critic network: μ', Q', with weights $\theta^{\mu^{'}} \leftarrow \theta^{\mu}$, $\theta^{Q'} \leftarrow \theta^{Q}$.
Initialize: Replay buffer D with capacity C, exploration counter.
Counter $= 0$

3: **for** step $t = 1$ to T **do**

4: **if** Counter $<$ C **then**

5: each USV i randomly chooses a_i;

6:

7: **else**

8: $a_i = \mu_{\theta_i}\left(o_{\text{new}}^{i}\right) + \mathcal{N}$

9: **end if**

10: Execute actions $a = [a_1, a_2, \ldots a_N]$, and observe reward R, new states o_{new};

11: Store transition $(o, a, r, o_{\text{new}})$ into experience replay buffer D;

12: Sample a mini-batch of β transitions $(o^m, a^m, r^m, o_{\text{new}}^m)$ from replay. Buffer D;

13: Set $y^m = R^m + \gamma Q_i^{\mu^{'}}\left(o_{\text{new}}^m, a_1', \ldots, a_N'\right)\Big|_{a_i'=\mu_i'(o_i)}$

14: Update critic by minimizing the loss

$$L(\theta_i) = \frac{1}{S}\sum_{m=1}^{S}\left(y^m - Q_i^{\mu}(o_{\text{new}}^m, a_1, \ldots, a_N)\right)^2$$

15: Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J = \frac{1}{S}\sum_{m=1}^{S} \nabla_{\theta_i}\mu_{\theta_i}\left(o_i^m\right)\nabla_{a_i} Q_i^{\mu}(o^m, a_1, a_2, \ldots, a_N)\Big|_{a_i=\mu_{\theta_i}(o_i^m)}$$

16: Updating actor networks
Updating critic networks
Update target networks with updating rate τ:
$\theta_i^{\mu^{'}} \leftarrow \theta_i^{\mu} + (1-\tau)\theta_i^{\mu^{'}}$
$\theta_i^{\mathbf{Q}^{'}} \leftarrow \theta_i^{\mathbf{Q}} + (1-\tau)\theta_i^{\mathbf{Q}^{'}}$

17: **end for**

18: **end for**

Now we resort to the MADDPG algorithm that extends DDPG to the multi-agent scenario. Specifically, a multi-agent game with L agents, $\mu_\theta = \left\{\mu_{\theta_1}, \ldots, \mu_{\theta_L}\right\}$, which is

parameterized by $\theta = \{\theta_1, \ldots, \theta_L\}$, can be regarded as the set of all deterministic policies for all agents, the gradient of the expected return for each agent is expressed as

$$\nabla_{\theta_i} J(\mu_{\theta_i}) = \mathbb{E}_{G,a\sim\mathcal{D}}\left[\nabla_{\theta_i}\mu_{\theta_i}(a_i \mid s_i) \times \nabla_{a_i} Q_i^{\mu}(o, a_1, a_2, \ldots, a_L)\Big|_{a_i=\mu_{\theta_i}(s_i)}\right] \tag{15}$$

where $Q_i^{\mu}(o, a_1, a_2, \ldots, a_L)$ is a centralized function with the inputs of all the agents' action $a = \{a_1, \ldots, a_L\}$, as well as the observation information o. $G = \{o_1, o_2, \ldots, o_L\}$ represents the information that the environment feeds back to the agent, including the distance between agents, the distance between agent i with surrounding obstacles, and the distance from agent i to each targets, all of which constitute the observation of all agents that are denoted by $\left[\mathrm{x}, \mathrm{y}, d_i^U, d_i^O, d_i^T\right]$. In our networks, the inputs of the critic network are the sates of all the USVs, the distance between USVs, as well as distance from the USV to the target area. The output of the critic network, i.e., $Q_i^{\mu}(o, a)$ is returned to the actor network for evaluating the action, which can be seen in Figure 2. The actor i executes the action based on its own policy μ_{θ_i}, and the critic uses the Q_i^{μ} function to evaluate the action of the actor. In order to remove the correlation of the samples generating from the environment, the framework adopts the experience replay mechanism to store the experience of each agent, which is a finite buffer with capacity M expressed as the tuple $(o, o', a_1, \ldots, a_L, r_1, \ldots, r_L)$. When the replay buffer is full, the oldest samples will be discarded. In order to better adapt to the environment and improve the stability of exploration, we updated the target network in a "soft update" way, so that it can slowly track the learning strategy, which can be expressed as $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$. In each time step, a batch size of tuples was selected to train the network, and the critic-network was updated by minimizing the loss function as

$$L(\theta_i) = \mathbb{E}_{\mathbf{o},\mathbf{o}',a,r}\left[\left(Q_i^{\mu}(o, a_1, \ldots, a_L) - y\right)^2\right] \tag{16}$$

where

$$y = r_i + \gamma Q_i^{\mu'}(o', a_1', \ldots, a_L')\Big|_{a_i'=\mu'(o_i')} \tag{17}$$

The update policy of the actor-network is expressed as

$$\nabla_{\theta_i} J \approx \frac{1}{m}\sum_k \nabla_{\theta_i}\mu_i\left(o_i^k\right)\nabla_{a_i} Q_i^{\mu}\left(o^k, a_1^k, \ldots, a_L^k\right)\Big|_{a_i=\mu_i(o_i^k)} \tag{18}$$

where m and k represent the minibatch size and the experience index, respectively.

4. Simulation Results

In this section, we present the simulation results as well as the performance analysis of the proposed algorithm. In Section 4.1, the settings for the simulation are developed. Section 4.2 presents the algorithm training configurations. In Section 4.3, we present the detailed figures for the reward, collision rate, and trajectories.

4.1. Environment Settings

We consider a 400 * 400 square training environment based on the open AI platform, which consists of USVs, targets, and threat areas. The geometric coordinate system is established with the environmental center as the origin of the coordinates. As the target area and static obstacle are regarded as parts of the environment, their settings would not be changed during the training process. In contrast, the dynamic obstacle changes position randomly after each training episode. In our settings, the maximum communication distance ρ_c is set to 0.9 and the width of critical area σ is set to 0.1. In this coordinate system, the size of the threat area is set to 0.25, while the size of the agent and target area are set to 0.04 and 0.20, respectively. We present two indicators to measure the effectiveness of

training: the collision rate of USVs' collisions during the current episode and the collision rate between the agent and the critical area during the total episode.

4.2. Training Configuration

In the given scenario, we set 60 episodes with 1000 steps each during the training process. As the mission is continuous, we manually aborted the training process in each training, and the environment was reset automatically after each episode. Since the capacity of the memory buffer we set was 3000, the training process did not start until the buffer was full, thus each USV adopted a random policy in the first 3000 steps, and then the whole network started to be trained in each step afterward.

The whole training networks of the mission planning for the AUV are constructed based on the PyTorch framework, where the training process is implemented on the NVIDIA GeForce RTX 3060 GPU. The critic network consists of five fully connected layers, including three hidden layers, of which the hidden units are set as 64, 52, and 30, respectively. The ReLU function is adopted as the non-linearity function in the networks. Specifically, the first hidden layer with 64 neurons obtains the concatenated vector of observations o and actions a, and the data then proceed through the following second and third hidden layers, with 52 and 30 neurons, respectively. The whole process is offline and can be divided into two parts: off-line centralized training and on-line decentralized execution. The advantage of using this method is that the system will have competitive execution speeds compared to the traditional methods during the testing stage, and the calculation could be gathered in the training stage.

4.3. Simulation Figures

Figure 3 shows the episode rewards obtained by USVs under different algorithms. The RL algorithm based on the policy gradient is used as the benchmark comparison algorithm. The experimental setting is for two USVs to perform navigation and assignment tasks with two target areas under the condition of no obstacles. It can be seen that the MADDPG has a significant advantage over the DDPG and REINFORCE algorithms, as it can get a higher reward, which proves the effectiveness of the proposed algorithm in solving multiple USVs' navigation and area allocation tasks.

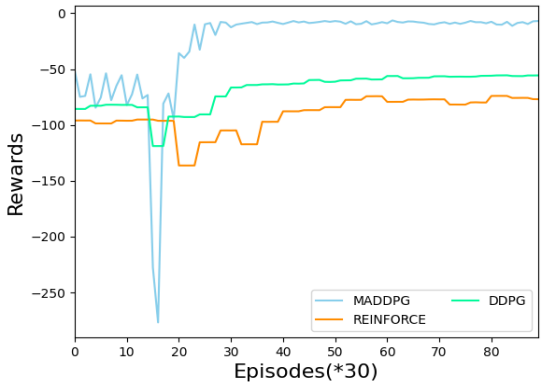

Figure 3. Rewards versus episodes under different RL algorithms.

Figure 4 shows the collision rate of USVs in different task environments. We observe the experimental effects when the number of USVs is 4, 3, and 2, and the number of target areas is 1, 2, 3, and 4, respectively. It can be seen from the figure that with the increase in training times, the collision rate of USVs in each turn will gradually decrease and tend to converge. At the same time, the total number of collisions will gradually decrease and stabilize, as can be seen in Figure 5.

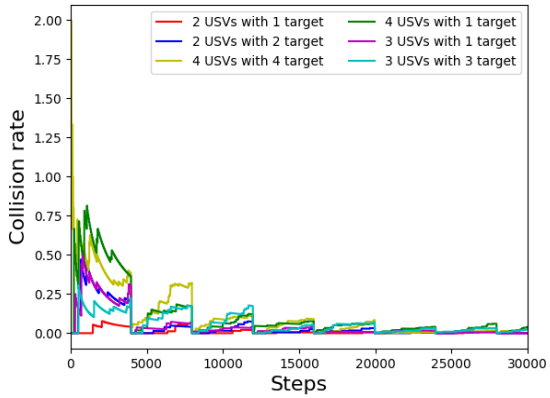

Figure 4. Collision rate per episode for the different numbers of USVs.

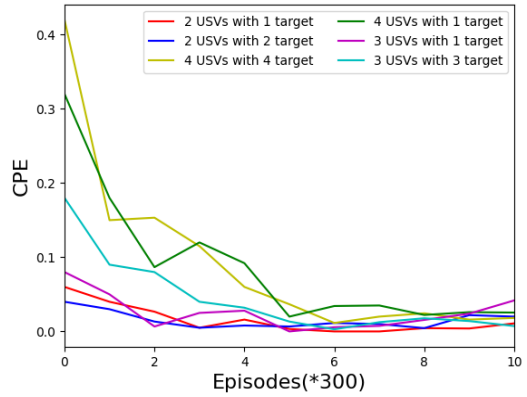

Figure 5. Collision rate in total episodes for the different numbers of USVs under the MADDPG algorithm.

Figure 6 shows the experimental conditions of the USV reward. It can be seen that in the early stage of the training, because of the blindness of the exploration stage, the reward for each USV will undergo a dramatic decline in the lowest (about 500 rounds), but with ongoing exploration, the neural network with gradually learn the best strategy.

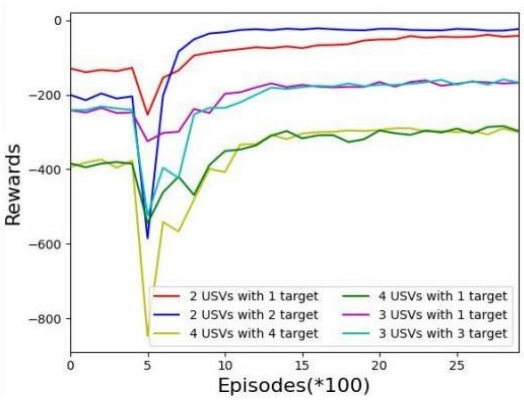

Figure 6. Rewards versus episodes for the different numbers of USVs under the MADDPG algorithm.

Figure 7 shows the dynamic trajectory of USVs in different test environments. Obstacles will move randomly after each training round, while the position of the target area remains unchanged. The number of USVs and target areas can be set manually. When encountering obstacles, it will adjust the direction according to the security of the surrounding environment and the position of the target area. When approaching the target area, it will automatically coordinate the area to be reached according to the position of each target area to complete the task assigned by the area.

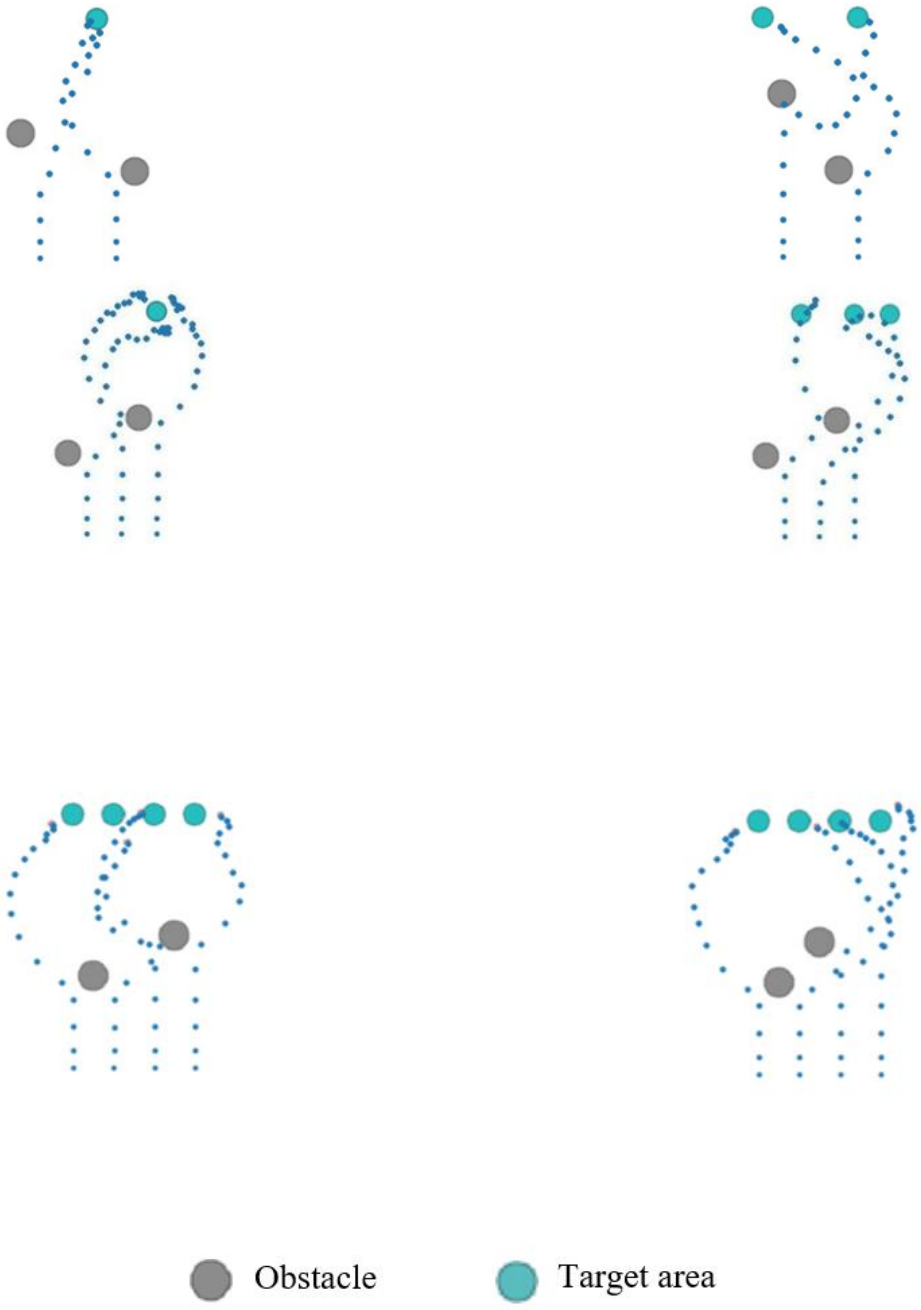

Figure 7. Dynamic trajectories of multi-USV.

5. Conclusions

In this paper, a multi-agent reinforcement learning framework is proposed to solve the trajectory design and area assignment for the multi-USV cooperation scenario. In this framework, each USV needs to select the closest target area so as to obtain the shortest route, during which each USV should keep a certain distance from others to keep within the communication distance and avoid collisions. All UAVs use the MADDPG to try to find the optimal policy to obtain a better reward, while traditional optimization methods can hardly handle the scenario with a large number of UAVs. Thus, the proposed method can be applied to the trajectory design and area assignment for multi-UAV scenarios. The simulation results show the effectiveness of the proposed framework.

Author Contributions: Conceptualization, J.W., S.L.; Data curation, J.W., S.L.; Methodology, J.W., S.L.; Writing—original draft, J.W.; Supervision, S.L., Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation of China (Grant 52271306); Innovative Research Foundation of Ship General Performance (Grant 31422120); Key Laboratory of Equipment Pre-Research Fund Project (6142215200106).

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Xiao, H.; Cui, R.; Xu, D. A Sampling-Based Bayesian Approach for Cooperative Multiagent Online Search With Resource Constraints. *IEEE Trans. Cybern.* **2017**, *48*, 1773–1785. [CrossRef] [PubMed]
2. Paull, L.; Saeedi, S.; Seto, M.; Li, H. Sensor-Driven Online Coverage Planning for Autonomous Underwater Vehicles. *IEEE/ASME Trans. Mechatron.* **2012**, *18*, 1827–1838. [CrossRef]
3. Wang, N.; Zhang, Y.; Ahn, C.K.; Xu, Q. Autonomous Pilot of Unmanned Surface Vehicles: Bridging Path Planning and Tracking. *IEEE Trans. Veh. Technol.* **2021**, *71*, 2358–2374. [CrossRef]
4. Qi, S.; Yao, P. Persistent Tracking of Maneuvering Target Using IMM Filter and DMPC by Initialization-Guided Game Approach. *IEEE Syst. J.* **2019**, *13*, 4442–4453. [CrossRef]
5. Vasilijević, A.; Nađ, Đ.; Mandić, F.; Mišković, N.; Vukić, Z. Coordinated navigation of surface and underwater marine robotic vehicles for ocean sampling and environmental monitoring. *IEEE-ASME Trans. Mechatron.* **2017**, *22*, 1174–1184. [CrossRef]
6. Wang, N.; Sun, J.C.; Er, M.J.; Liu, Y.C. A novel extreme learning control framework. of unmanned surface vehicles. *IEEE Trans. Cybern.* **2015**, *46*, 1106–1117. [CrossRef]
7. Kim, H.; Park, B.; Myung, H. Curvature path planning with high resolution graph. for unmanned surface vehicle. In *Robot Intelligence Technology and Applications 2012*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 147–154.
8. Soulignac, M. Feasible and Optimal Path Planning in Strong Current Fields. *IEEE Trans. Robot.* **2010**, *27*, 89–98. [CrossRef]
9. Chen, M.Z.; Saad, W.; Yin, C.C. Liquid State Machine Learning for Resource and Cache Management in LTE-U Unmanned Aerial Vehicle (UAV) Networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1504–1517. [CrossRef]
10. Zeng, Z.; Lammas, A.; Sammut, K.; He, F.P.; Tang, Y.H. Shell space decomposition based path planning for AUVs operating in a variable environment. *Ocean Eng.* **2014**, *91*, 181–195. [CrossRef]
11. Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; How, J.P. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.* **2009**, *17*, 1105–1118. [CrossRef]
12. Karaman, S.; Walter, M.R.; Perez, A. Any time motion planning using the RRT*. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1478–1483.
13. Salzman, O.; Halperin, D. Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning. *IEEE Trans. Robot.* **2016**, *32*, 473–483. [CrossRef]
14. Zhang, Y.; Chen, J.; Shen, L. Real-time trajectory planning for UCAV air-to-surface attack using inverse dynamics optimization method and receding horizon control. *Chin. J. Aeronaut.* **2013**, *26*, 1038–1056. [CrossRef]
15. Ma, C.S.; Miller, R.H. MILP optimal path planning for real-time applications. In Proceedings of the American Control Conference, Minneapolis, MN, USA, 14–16 June 2006; pp. 4945–4950.
16. Fayazi, S.A.; Vahidi, A. Mixed-Integer Linear Programming for Optimal Scheduling of Autonomous Vehicle Intersection Crossing. *IEEE Trans. Intell. Veh.* **2018**, *3*, 287–299. [CrossRef]
17. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1986; pp. 396–404.
18. Yao, P.; Qi, S. Obstacle-avoiding path planning for multiple autonomous underwater vehicles with simultaneous arrival. *Sci. China Technol. Sci.* **2018**, *62*, 121–132. [CrossRef]
19. Xie, S.R.; Wu, P.; Peng, Y.; Luo, J.; Qu, D.; Li, Q.M.; Gu, J. The obstacle avoidance planning of USV based on improved artificial potential field. In Proceedings of the 2014 IEEE International Conference on Information and Automation (ICIA), Hulunbuir, China, 28–30 July 2014; pp. 746–751.
20. Wang, N.; Karimi, H.R. Successive Waypoints Tracking of an Underactuated Surface Vehicle. *IEEE Trans. Ind. Inform.* **2019**, *16*, 898–908. [CrossRef]
21. Ma, Y.; Hu, M.; Yan, X. Multi-objective path planning for unmanned surface vehicle with currents effects. *ISA Trans.* **2018**, *75*, 137–156. [CrossRef] [PubMed]
22. Wang, J.K.; Chi, W.Z.; Li, C.M.; Wang, C.Q.; Meng, M.Q.H. Neural RRT*: Learning-Based Optimal Path Planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758. [CrossRef]
23. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. In Proceedings of the International Conference on Learning Representations 2016, San Juan, Puerto Rico, 2–4 May 2016. arXiv:1511.05952v4.

24. Sang, H.Q.; You, Y.S.; Sun, X.J.; Zhou, Y.; Liu, F. The hybrid path planning algorithm. based on improved A* and artificial potential field for unmanned surface vehicle formations. *Ocean Eng.* **2020**, *223*, 108709. [CrossRef]
25. Wang, N.; Jin, X.Z.; Er, M.J. A multilayer path planner for a USV under complex. marine environments. *Ocean. Eng.* **2019**, *184*, 1–10. [CrossRef]
26. Wang, N.; Xu, H.W. Dynamics-Constrained Global-Local Hybrid Path Planning of. an Autonomous Surface Vehicle. *IEEE Trans. Veh. Technol.* **2020**, *69*, 6928–6942. [CrossRef]
27. Gu, Y.; Zhang, Y.; Rong, Z.W.; Tong, H.Z. Unmanned Surface Vehicle Collision Avoidance Path Planning in Restricted Waters Using Multi-Objective Optimisation Complying with COLREGs. *Sensors* **2022**, *22*, 5796. [CrossRef] [PubMed]
28. Wu, Y.; Low, K.H.; Lv, C. Cooperative Path Planning for Heterogeneous Unmanned Vehicles in a Search-and-Track Mission Aiming at an Underwater Target. *IEEE Trans. Veh. Technol.* **2020**, *69*, 6782–6787. [CrossRef]
29. Pradhan, B.; Nandi, A.; Hui, N.B.; Roy, D.S.; Rodrigues, J.J.P.C. A Novel Hybrid Neural Network-Based Multirobot Path Planning With Motion Coordination. *IEEE Trans. Veh. Technol.* **2020**, *69*, 1319–1327. [CrossRef]
30. Wu, Y.; Low, K.H.; Pang, B.Z.; Tan, Q.Y. Swarm-Based 4D Path Planning For Drone. Operations in Urban Environments. *IEEE Trans. Veh. Technol.* **2021**, *70*, 7464–7479. [CrossRef]
31. Guo, X.H.; Ji, M.J.; Zhao, Z.W.; Wen, D.S.; Zhang, W.D. Global path planning and. multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (PSO) algorithm. *Ocean Eng.* **2020**, *216*, 107693. [CrossRef]
32. Nazarahari, M.; Khanmirza, E. Multi-objective multi-robot path planning in. continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
33. Zhong, C.; Yao, J.; Xu, J. Secure UAV Communication With Cooperative Jamming and Trajectory Control. *IEEE Commun. Lett.* **2018**, *23*, 286–289. [CrossRef]
34. Ye, H.; Li, G.Y.; Juang, B.-H.F. Deep Reinforcement Learning Based Resource Allocation for V2V Communications. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3163–3173. [CrossRef]
35. Cao, Z.G.; Guo, H.L.; Song, W.; Gao, K.Z.; Chen, Z.H.; Zhang, L.; Zhang, X.X. Using reinforcement learning to minimize the probability of delay occurrence in transportation. *IEEE Trans. Veh. Technol.* **2020**, *69*, 2424–2436. [CrossRef]
36. Xiang, H.; Peng, M.; Sun, Y.; Yan, S. Mode Selection and Resource Allocation in Sliced Fog Radio Access Networks: A Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4271–4284. [CrossRef]
37. Zhou, X.; Wu, Q.; Yan, S.; Shu, F.; Li, J. UAV-Enabled Secure Communications: Joint Trajectory and Transmit Power Optimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4069–4073. [CrossRef]
38. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6382–6393.
39. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Eleventh International Conference on International Conference on Machine Learning, New Brunswick, NJ, USA, 10–13 July 1994; pp. 157–163.