sensors

MDPI

Article

Path Planning for Unmanned Surface Vehicles with Strong Generalization Ability Based on Improved Proximal Policy Optimization

Pengqi Sun [1,2], Chunxi Yang [1,2], Xiaojie Zhou [1,2] and Wenbo Wang [1,2,*]

[1] Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; spq5597@stu.kust.edu.cn (P.S.); ycx@kmust.edu.cn (C.Y.); zxj984256@stu.kust.edu.cn (X.Z.)
[2] Yunnan International Joint Laboratory of Intelligent Control and Application of Advanced Equipment, Kunming 650500, China
* Correspondence: wenbo_wang@kust.edu.cn

Abstract: To solve the problems of path planning and dynamic obstacle avoidance for an unmanned surface vehicle (USV) in a locally observable non-dynamic ocean environment, a visual perception and decision-making method based on deep reinforcement learning is proposed. This method replaces the full connection layer in the Proximal Policy Optimization (PPO) neural network structure with a convolutional neural network (CNN). In this way, the degree of memorization and forgetting of sample information is controlled. Moreover, this method accumulates reward models faster by preferentially learning samples with high reward values. From the USV-centered radar perception input of the local environment, the output of the action is realized through an end-to-end learning model, and the environment perception and decision are formed as a closed loop. Thus, the proposed algorithm has good adaptability in different marine environments. The simulation results show that, compared with the PPO algorithm, Soft Actor–Critic (SAC) algorithm, and Deep Q Network (DQN) algorithm, the proposed algorithm can accelerate the model convergence speed and improve the path planning performances in partly or fully unknown ocean fields.

Keywords: USV; path planning; deep reinforcement learning; deep neural network; generalization; perception

check for updates

Citation: Sun, P.; Yang, C.; Zhou, X.; Wang, W. Path Planning for Unmanned Surface Vehicles with Strong Generalization Ability Based on Improved Proximal Policy Optimization. *Sensors* **2023**, *23*, 8864. https://doi.org/10.3390/s23218864

Academic Editor: Sergio Toral Marín

Received: 29 September 2023
Revised: 18 October 2023
Accepted: 25 October 2023
Published: 31 October 2023

Copyright: © 2023 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

In recent years, there has been an increasing global emphasis on the importance of marine resources, coinciding with the rapid advancement of artificial intelligence. In this context of technological convergence, Unmanned Surface Vehicles (USVs) have gained considerable traction in various fields including scientific research, ocean resource exploration, water rescue missions, and environmental initiatives [1–4]. Given the inherently complex and dynamic nature of the marine environment, effective path planning for USVs plays a crucial role in ensuring the successful execution of the aforementioned tasks.

The navigation system of Unmanned Surface Vehicles (USVs) comprises three major subsystems: environmental and navigation situation awareness, cognition-based decision making, and ship navigation control. Path planning and obstacle avoidance are fundamental challenges in constructing these subsystems [5]. A typical path planning task aims to provide collision-free navigation from the starting position to the specified target position on a given map or grid [6,7]. Currently, USV path planning and obstacle avoidance techniques can be broadly classified into traditional methods and intelligent methods. Traditional methods typically refer to deterministic approaches [8] that provide solutions following predefined rules, using fused information at each decision step. Among traditional methods, Iijima et al. [9] used the width-first search method to select and plan

collision avoidance paths. However, their approach did not consider the influence of the navigation environment. Churkin et al. [10] attempted to establish a mathematical model for collision avoidance path planning using both continuous and discrete research methods. However, the continuous method exhibited high computational complexity and was unsuitable for cases involving multiple USV encounters. In another study, Hwang et al. [11] employed the fuzzy set theory to establish a knowledge base system to evaluate ship collision risk and determine collision avoidance strategies. However, their system solely focused on collision avoidance strategies, rendering the suboptimal overall voyage. Chang et al. [12] proposed a model for calculating collision avoidance paths on grid maps using a maze routing algorithm, but this approach did not account for international maritime collision avoidance rules and navigation environment conditions.

Szlapcynski et al. [13] improved the maze routing method in [12] by adding a turning penalty and a time-varying restricted area. However, the resulting calculated path remained suboptimal due to the neglect of navigation environment conditions.

Apart from the aforementioned modeled based methods, a number of heuristic algorithms have also been proposed. Recently, a novel Voronoi–Visibility path planning algorithm, which integrates the advantages of a Voronoi diagram and a Visibility graph, was proposed for solving the USV path planning problem in [14]. In [15], Nie et al. studied the problem of robot path planning using the Dijkstra algorithm and Ant Colony Optimization. For known environments, the path planning problem was studied in [16], which introduced geometric areas to divide obstacle avoidance zones and perform global obstacle-avoidance planning of USV with an improved A-star algorithm. In [17], Yao et al. proposed a hierarchical architecture using the biased min-consensus method to address the path planning problem of USVs. In [18], Wu et al. investigated USV path planning by proposing a global path planning approach for USVs using an intelligent water drops algorithm. Wei et al. [19] designed a trajectory planning unit based on the unique characteristics of USVs, reflecting the intelligent navigation of USVs. The aforementioned methods demonstrate a wide application prospect in the field of USVs.

However, determining the optimal obstacle-avoidance path for USVs involves a number of crucial factors, including navigation environmental conditions and international maritime collision-avoidance rules. Many of these factors are abstract and qualitative, making them challenging to quantify using deterministic mathematical methods. In contrast, intelligent methods, such as Deep Reinforcement Learning (DRL) algorithms, show better efficacy in handling abstract and qualitative influencing factors, making them more suitable for USV path planning and obstacle avoidance in uncertain and time-varying ocean environments. DRL combines the feature interpretation capabilities of deep learning with the decision-making abilities of reinforcement learning, enabling direct optimal decision outputs based on high dimensional input data. This approach constitutes an end-to-end decision control system [20,21]. Trained with DRL, Jaradat et al. [22] incorporated a predictive model into DRL, achieving high dynamic performance in convergence speed, average reward value, and other indicators through path planning experiments on aircraft carrier decks. Guan et al. [23] proposed a local path planning and behavior decision-making approach based on an improved Proximal Policy Optimization (PPO) algorithm, enabling smart USVs to reach their targets without requiring human experience. To further enhance ship path planning during navigation, Guo et al. [24] introduced a coastal ship path planning model based on the Deep Q-Network (DQN) algorithm. Prianto et al. [25] developed a path planning algorithm based on Soft Actor–Critic (SAC), allowing for multi-arm manipulator path planning.

Convolutional layers have been widely applied to the feature extraction problem of high-dimensional state tasks in DRL. Habib et al. [26] gave detailed insight into computation acceleration using stochastic gradient descent, fast convolution, and parallelism in CNN. Lebedev et al. [27] covered the approaches based on tensor decompositions, weight quantization, weight pruning, and teacher–student learning. Krichen [28] provided a comprehensive overview of CNNs and their applications in image recognition tasks.

In this paper, a path planning algorithm of USV with local environmental information perception for a time-varying maritime environment is proposed based on an improved Proximal Policy Optimization (PPO) algorithm. The contributions of this study can be summarized in the following three key aspects:

- To reflect realistic maritime environments, a grid-based environment model is constructed based on real-world electronic charts to map the dynamic states of a ship and static obstacles in the sea.
- Integration of planning and obstacle avoidance is achieved based on the proposed PPO algorithm, with the consideration of the sensing range of on-board sensors.
- To address the unpredictable situations, e.g., unknown maps or moving ships in the area, we use convolutional neural networks (CNNs) for state-feature extraction in PPO. Our simulation results show that this method greatly improves the adaptability of USV in path planning in uncharted marine environments.

The rest of this paper is organized as follows. The problem formulation is described in Section 2. In Section 3, a path planning algorithm based on PPO is proposed. Section 4 presents a comparative analysis of the simulation experiment process and experimental results. Finally, the conclusion and future work are given in Section 5.

2. Modeling and Problem Formulation

To accurately represent the significant environmental characteristics of the sea, this study adopts the raster model proposed by [29], using electronic nautical charts. The marine environment model includes the raster model, physical model, numerical model, and intelligent model. The raster model has the characteristics of high spatial resolution, data analysis, and visualization. These characteristics enable the model to better capture the detailed features (e.g., coastline and seabed topography) in different areas of the ocean. The output data generated by the grid model can be analyzed and visualized in various forms. Thus, the raster model and electronic nautical charts are leveraged to transform both stationary and dynamic obstacles within a specified sea area into their corresponding raster representations.

2.1. Building a Marine Environment Map Based on Electronic Charts

The detailed information about the marine environment is extracted from the electronic chart. Firstly, a specific local electronic chart is selected from the global electronic chart, as exemplified in Figure 1. In this example, the chosen chart corresponds to the actual water area of the customized No. 1 warning area within the Ningbo–Zhoushan core port region (regional range: 122.106833° E∼122.286833° E, 29.817667° N∼29.893333° N). Subsequently, grid maps representing the relevant maritime area are generated through several steps, including land information extraction and binary grid processing. Further details regarding the specific grid map relevant to the electronic chart is provided on the right-hand side of Figure 1. Such an approach of grid processing retains the outlines of obstacles (such as lands and islands). Static obstacles such as highways and anchor points can be further added to the grid according to maritime traffic rules. By doing so, reliable information is provided to construct a high-accuracy grid model of the sea environment of interest.

2.2. Problem Formulation

The diagram that illustrates the USV and the definition of the sensing radius is presented in Figure 2a, where x_a and y_a represent the coordinates in the earth coordinate system, and x_{usv} and y_{usv} represent the coordinates in the ship coordinate system. In this study, the USV sensing radius is denoted by r_{sd}, and the white grid and black grid are defined as A_{free} and A_{obs}, respectively. The white grid A_{free} corresponds to the feasible area, while the black grid A_{obs} represents the obstacles such as land, ships, and other obstacles on the sea surface.

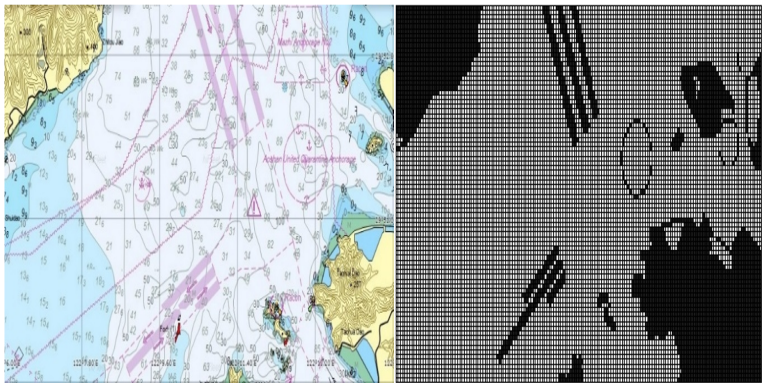
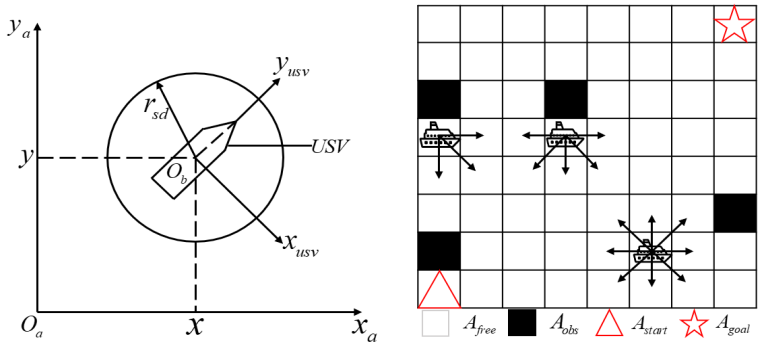


Figure 1. Electronic nautical charts of a given area [4].



(a) USV schematic and sensing radius (b) USV path planning map

Figure 2. USV schematic and sensing radius. The starting position A_{start} = (1, 1), the goal position A_{goal} = (8, 8), A_{free} is the position that can be passed, and A_{obs} is the obstacle. This paper aims to plan the safe path of USV from the starting position to the goal position.

In this study, we discretize the action space of a USV into a finite set. As illustrated in Figure 2b, when no obstacles are present around the USV, it is able to move in eight adjacent directions within the grid representation of the sea environment. Additionally, when it is at the edge of the finite grid map, all available movement directions lead towards the interior of the map.

3. USV Path Planning Based on PPO

3.1. State Space

To generalize the real-world application scenarios, we assume in this paper that an environment sensor (e.g., an ultra wide-band sensor) is deployed on the USV for sea environment detection. The detection range of the radar sensor is set according to the mapping granularity of the raster environment model. For instance, Figure 3 shows a sensing range of unit length on the grid map as given in Figure 1, which corresponds to the actual length of 102 m at sea. In this paper, we adopt the USV parameters from the literature [20], where a USV performs path planning at a sailing speed of 9 nautical miles per hour. As illustrated in Figure 3, we consider the shaded part in the figure as the detection range of 360°, which returns the coordinate information of all obstacles within the detection range of the sensor. We also assume that the real-time coordinate information of the USV is available with an on-board GPS sensor.

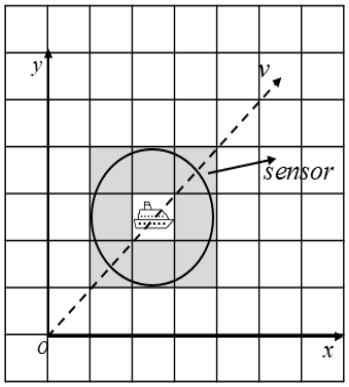


Figure 3. Radar sensor detection range.

A sensed environment state of the USV is composed of three feature elements: the USV's own coordinates, the perceived obstacle coordinate information organized as a vector, and the USV's current distance from the destination. Then, the state vector of the USV, s, can be expressed by concatenating the aforementioned three vectors as follows:

$$s = [(x, y)_{usv}, M, d]^{\mathrm{T}}, \tag{1}$$

where M represents the sensing information within the sensor's perceptual range, and $(x, y)_{usv}$ is the real-time coordinates of the USV. The Euclidean distance between the USV and the destination is shown in (2):

$$\mathrm{d} = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2}. \tag{2}$$

In (2), x_{usv}, y_{usv} represent the USV coordinates, and x_{goal}, y_{goal} represent the destination coordinates. The environment sensing results regarding the obstacles are encoded into a binary image, where, following the rule of grid map generation, the white cells represent the available area and black cells indicate the position of obstacles.

3.2. Action Space

We assume that the USV maintains a constant cruising speed measured in nautical miles per hour. Accordingly, we define the action set of the USV as consisting of eight discretized moving directions. Without loss of generality, we consider that, at each time slot, the USV can move from its current grid to any of the adjacent eight grids, as shown in Figure 4. In our algorithm, we use one-hot coding to represent the eight possible actions a.

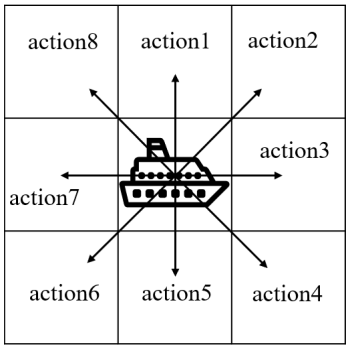


Figure 4. Actions of the USV.

3.3. Reward Function

We consider that our objective is to maximize the expected cumulative discounted reward along time for the USV. Therefore, the design of the instantaneous reward func-

tion is the key factor to determine the way of strategy adapation in the reinforcement learning process.

$$r(s,a) = r_1(s,a) + r_2(s,a) + r_3(s,a) \tag{3}$$

where s and a are defined in Sections 2.1 and 2.2, which represent state space and action space, respectively.

With our algorithm design, the instantaneous reward function consists of three parts: the reward for the USV's distance from the destination r_1, the penalty for collision with obstacles r_2, and the one-time reward/penalty for arriving at the destination or moving out of bounds r_3.

We define r_1 as follows:

$$r_1(s,a) = \begin{cases} 5 & \text{if } d_t < d_{t-1}, \\ -7 & \text{if } d_t \geq d_{t-1}, \end{cases} \tag{4}$$

where d_t represents the distance (see (2)) from the destination at time slot t. (4) indicates that, when the USV is closer to the target position after taking a one-step action, the learning algorithm will give it an appropriate reward. Otherwise, a penalty will be imposed. Comparatively, r_2 gives the penalty for collisions with obstacles:

$$r_2(s,a) = \begin{cases} -5 & \text{if } \mathbb{I}((x,y)_{usv}(s,a), \text{ob}), \\ -0.002 & \text{if } \textit{otherwise}, \end{cases} \tag{5}$$

where $\mathbb{I}((x,y)_{usv}, \text{ob})$ is the indicator function showing whether the designated USV position collides with an obstacle. Notice that we introduce a small penalty for the situation of the USV remaining idle on the map. This method encourages the USV to reach its destination as soon as possible.

Finally, $r_3(s,a)$ indicates that the USV is either rewarded upon successful completion when it reaches the target location or incurs a failure penalty if its coordinates exceed the designated boundary or if collisions occur:

$$r_3(s,a) = \begin{cases} 100, \text{ if } d_t \leq 0.1, \\ -5, \text{ if } \mathbb{I}((x,y)_{usv}(s,a), \text{ob}) \text{ or } \mathbb{I}((x,y)_{usv}(s,a), \text{ out-b}), \end{cases} \tag{6}$$

where $\mathbb{I}((x,y)_{usv}, \text{out-b})$ indicates whether the USV is out of the map boundary. With (5) and (6), when the USV reaches the target position, a large one-time reward is given. When the USV is out of boundary or collides with an obstacle, a penalty of -5 is given at task termination.

3.4. Improved PPO with Better Generalization Capability

In this section, we propose to adapt the PPO algorithm in two stages to solve the path planning problem in USV.

3.4.1. Neural Network Design with Convolutional Layers

We employ convolutional layers to extract the feature of the obstacle map obtained from the sensing results. The calculation of the i-th feature graph of the n-th convolutional layer is defined as follows:

$$x_i^n = f\left(0, \sum_{j \in M_i} x_j^{n-1} k_{ji}^n + b_i^n\right), \tag{7}$$

where M_i is the set of feature graphs, for which the initial input is given in (7), k_{ji}^n is the i-th convolution kernel of the n layer, b_i^n is the i-th configuration of the n layer, and $f(\cdot)$ is the activation function, for which we use the ReLU function in our designed neural network. For our adopted PPO neural network, the details of the network parameters are given in Table 1.

Table 1. Parameters of the actor–critic networks for PPO with convolution layers.

Model Network	Input	Kernel-Size	Fc	Activation Function	Output
Actor	state-dim	$3 \times 3, 1, 1$	$16 \times 16 \times 45$	Relu	action-dim
Critic	state-dim	$3 \times 3, 1, 1$	$16 \times 16 \times 45$	Relu	status-value

3.4.2. PPO Algorithm

Proposed by OpenAI [30], the PPO algorithm introduces a new objective function that can be updated in small batches with multiple training steps. This solves the problem where the step size is difficult to determine as in the vanilla Policy Gradient (PG) algorithm. As one PG algorithm, the main idea of PPO is to use gradients to update the USV's strategy $\pi(a|s)$ in order to maximize the expected cumulative reward. In a PG algorithm, the objective function of the network parameter θ update is as follows:

$$L^{PG}(\theta) = \mathrm{E}_t[\log \pi_\theta(a_t|s_t)A_t], \tag{8}$$

where A_t is the advantage function estimating the value of taking action a_t in state s_t compared to the average expected return. A_t will be given in our later discussion.

The PG algorithm is highly sensitive to the step size, making it challenging to select an appropriate value. To overcome this limitation, PPO uses the ratio of the action probability $\pi_\theta(a_t|s_t)$ under the current strategy to the action probability $\pi_{\theta_{old}}(a_t|s_t)$ of the previous strategy to observe the effect of the agent's action. The ratio of old and new strategies is recorded as

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{9}$$

If the ratio between the new and old strategy functions is $r_t(\theta) > 1$, it indicates that the probability of the action occurring under this policy is higher than that of the previous policy. Otherwise, the probability is lower than the previous policy. The objective function used for training is given as follows:

$$L^{CPI}(\theta) = \mathrm{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right] = \mathrm{E}_t[r_t(\theta)A_t], \tag{10}$$

where A_t is the advantage function:

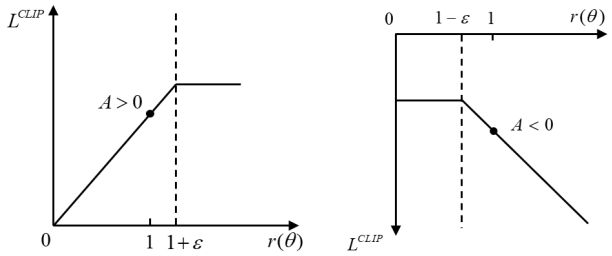
$$A_t = \mathrm{Q}(\mathrm{s}_t,\mathrm{a}_t) - \mathrm{V}(\mathrm{s}_t,\mathrm{a}_t) \tag{11}$$

In (11), $\mathrm{Q}(\mathrm{s}_t,\mathrm{a}_t)$ is the cumulative reward value of taking action a_t under state s_t, and $\mathrm{V}(\mathrm{s}_t,\mathrm{a}_t)$ is the estimated state value. $A_t > 0$ indicates that the current action is better than the average action and the learning process increases the probability of choosing the action. Otherwise, when $A_t < 0$, it means that the action is worse than the average action and the probability of choosing the action will be reduced.

The PPO algorithm improves the stability of training agent behaviors in PG by constraining policy updates to a small range. The objective function of PPO is improved from (10) as

$$L^{CLIP}(\theta) = \mathrm{E}_t[\min(r_t(\theta)A_t, CLIP(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)], \tag{12}$$

where the clip function is a truncation function that limits the value of the old and new policy parameters $r(\theta)$ to the interval $[1-\varepsilon, 1+\varepsilon]$. ε is a truncation constant used to assist in setting the range of policy updates; it is usually set to 0.1 or 0.2. Figure 5a,b illustrate these two situations of truncation [30], respectively. The pseudo code for the PPO–Clip algorithm is given in Algorithm 1.



(a) $1+\varepsilon$ truncation (b) $1-\varepsilon$ truncation

Figure 5. Restricted range of objective function.

Algorithm 1: Pseudo code of the PPO–Clip algorithm

1: Input: initial policy parameters θ_0, value function parameters ϕ_0, clipping threshold ϵ.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $D_k = \tau_i$ by running policy $\pi_k = \pi(\theta_k)$.
4: Compute advantage estimates, $A_t^{\pi_k}$ (using any method of advantage estimation) based on the current value function $V_{\phi k}$.
5: Update the policy by maximizing the PPO–Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} L_{\theta_k}^{CLIP}(\theta)$$

typically via stochastic gradient ascent with Adam, where

$$L_{\theta_k}^{CLIP}(\theta) = \underset{\tau \sim \pi_k}{E}\left[\sum_{t=0}^{T}\left[\min(\mathrm{r}_t(\theta)A_t^{\pi_k}, clip(\mathrm{r}_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\pi_k})\right]\right]$$

6: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_\phi(s_t) - R_t)^2$$

typically via some gradient descent algorithm.
7: **end for**

4. Experiments

In this section, we provide a series of numerical simulation results to evaluate the performance of our proposed algorithm. For marine environment simulation and USV strategy training, all the experiments are performed with Pytorch 2.0.1 on a desktop machine with 128 G memory and hardware acceleration using a GeForce RTX 3090Ti GPU from NVIDIA Santa Clara, CA, USA. We aim to validate the generalizability of the proposed algorithm by modifying three conditions: endpoint coordinates, map, and the number of training sets. We conduct simulation experiments from various aspects to verify the effectiveness of our approach. Experiment 1 focuses on a USV obstacle avoidance simulation using the algorithm proposed in this paper. In Experiment 2, we test the generalization capability of the proposed algorithm by changing the endpoint. Similarly, in Experiment 3, we explore the algorithm's generalization under different sea maps. Experiment 4 involves training an additional network model by increasing the number of maps used for training. This model is then used to assess the generalization of the proposed algorithm in the simulation environment of Experiment 2. Finally, Experiment 5 involves comparing the performance of the proposed algorithm with other algorithms, thereby demonstrating its effectiveness.

4.1. Generalization Definition and Modeling

For reinforcement learning (RL), the generalization ability refers to when the reinforcement learning model is trained in the training environment and the performance of

the model is verified in the test environment for the same task in the same domain. In supervised learning, some predictor is trained on a training dataset, and the performance of the model is measured on a held out testing dataset. It is often assumed that the data points in both the training and testing dataset are drawn independently and identically distributed from the same underlying distribution. The generalization gap in supervised learning for a model ϕ with training and testing data D_{train}, D_{test} and loss function L is defined as

$$GenGap(\phi) := \mathrm{E}_{(x,y)\sim D_{test}}[L(\phi, x, y)] - \mathrm{E}_{(x,y)\sim D_{train}}[L(\phi, x, y)]. \tag{13}$$

This gap is used as a measure of generalization, specifically, a smaller gap means a model generalizes better. Generalization refers to a class of problems, rather than a specific problem. Thus, the generalization measure in RL is shown in Equation (13).

To discuss generalization, we need a way of talking about a collection of tasks, environments, or levels; the need for generalization emerges from the fact that we train and test the policy on different collections of tasks. To formalize the notion of a collection of tasks, we start with the Contextual Markov Decision Process (CMDP). The CMDP is shown as Equation (14):

$$M = (S', A, O, R, T, C, \phi, p(S'|C), p(C)). \tag{14}$$

where S' is the underlying state space; A is the action space; O is the observation space; R is the scalar reward function; T is the Markovian transition function; C is the context space; and $\phi : S' \times C \to O$ is the emission or observation function. We factorize the initial state distribution as shown in (15):

$$p(S', C) = p(C)p(S'|C). \tag{15}$$

and we call $p(C)$ the context distribution. This distribution is what is used to determine the training and testing collections of levels, tasks, or environments. We now describe the class of generalization problems we focus on, using the CMDP formalism.

All else being equal, the more similar the training and testing environments are, the smaller the generalization gap and the higher the test time performance. The categorization of the methods for tackling generalization in RL is shown in Figure 6.

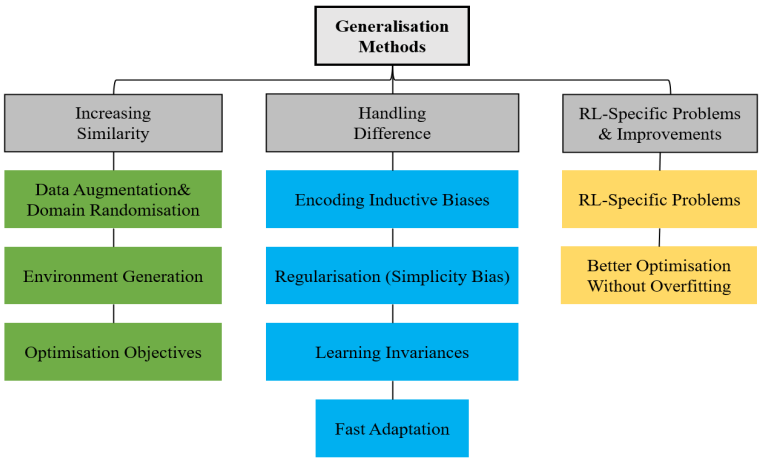


Figure 6. Categorization of methods for tackling generalization in reinforcement learning [31].

In this paper, the generalization of the model is verified by holdout validation and data augmentation. For the holdout validation method, the map set is divided into the training set and the test set. The training set specifically refers to the simulation data in Experiment 1. The test set refers to the simulation data in Experiment 2 and Experiment 3. The results of Experiment 2 and Experiment 3 show that the USV can successfully reach the end point under different test sets. For the data augmentation method, by transforming and expanding the training data, more samples are introduced. Then, Experiment 4 tests

the generalization of the model under the condition of multiple training data sets by using the simulation environment of Experiment 2.

4.2. Simulation Experiment

4.2.1. Experimental Platform Description and Training Parameters

The USV simulation environment in this paper is shown in Figure 7, which is extracted from the data as illustrated in Figure 1. In this environment, the USV is represented by the blue square in the figure. The gray squares represent obstacles. The yellow squares represent the end of the path. USV continuously learns and explores its strategy, following the method proposed in this paper. The entire scene is reset at the end of each round or when the maximum number of time steps is exceeded in a single round. The detection radius of the radar is set to 153 m. In the simulation experiment, the performance indicator values include the iterations used by the USV to reach the end point, the time needed for algorithm convergence, and the average reward obtained. Among them, the step numbers can be reflected by the path diagram of each experiment. The allowable number of the training time steps in this experiment is 3×10^5, and the PPO parameter settings are shown in Table 2. The convergence time and average reward are shown in Table 3.

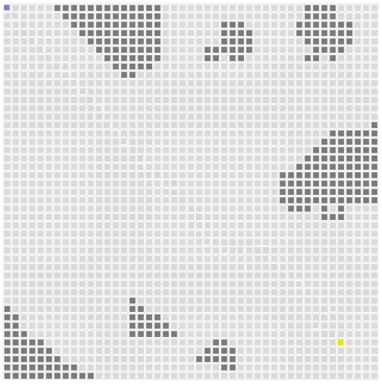


Figure 7. USV simulation in the sea environment.

Table 2. Hyperparameters for the proposed PPO algorithm.

Parameters	Value
Actor learning rate	1×10^{-4}
Critic learning rate	3×10^{-4}
Discount factor	0.99
Batch size	1500
Time steps	3×10^5
Clip range	0.2

4.2.2. Experimental Results and Analysis

For Experiment 1, the path trace following the proposed algorithm and the convergence tendency of the average reward are shown in Figures 8 and 9, respectively. We also record the snapshot of the paths for the USV to reach the destination in each episode, as shown in Figure 8. The average reward converges when the episode reaches 341. During the training process, with the improvement of the strategy, the number of steps is reduced when the USV reaches the end at each episode. To find a better strategy, the PPO algorithm explores the unknown action space. Therefore, when the episode reaches 1634, the number of steps is increased.

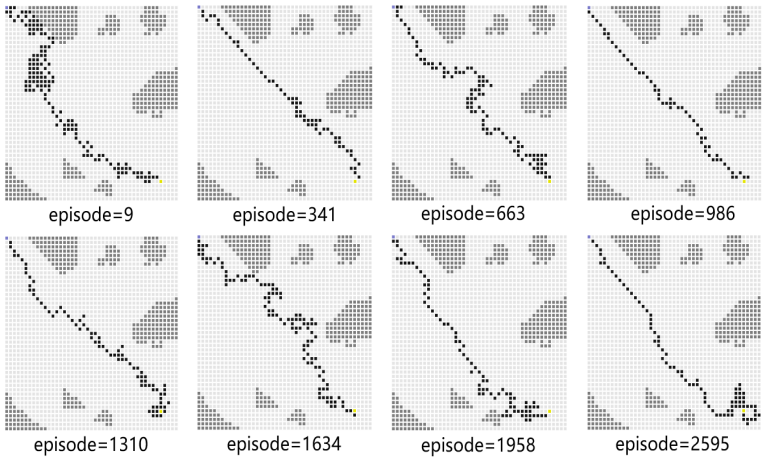


Figure 8. The planned path for Experiment 1.

As shown in Figure 9, as the iteration time steps increase, the average reward converges when the algorithm iterates to roughly 10^5 time steps. The final converged reward fluctuates between -111.34 and -125.55, which shows that the proposed algorithm is basically converging.

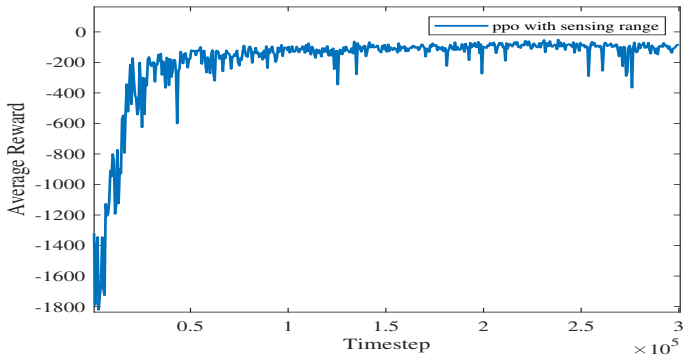


Figure 9. The average reward evolution for Experiment 1.

In Figure 10, the vanilla PPO algorithm with zero sensing range is used for path planning under the same environment. It can be seen from the figure that PPO using convolutional layers achieves a better cumulative reward. In the absence of a sensing range, the USV is not able to handle different obstacle environments (see the near-to-end stage of the training process). The experimental results show that increasing the sensing range can greatly improve the convergence efficiency of USV as well as the obstacle-handling capability.

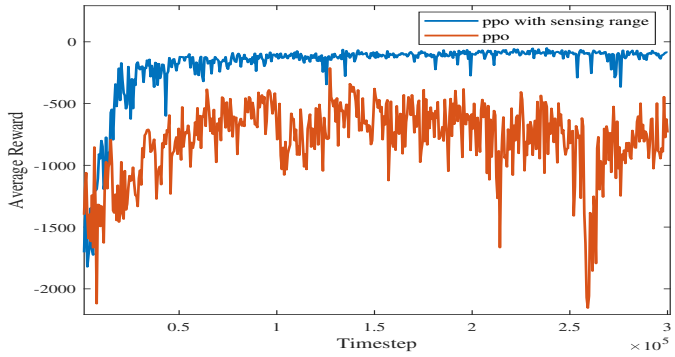


Figure 10. Comparison of training processes with different methods using PPO.

In Experiment 2, the generalization capability of the proposed algorithm is verified by modifying the path endings. The starting point coordinates of the simulation environment remain unchanged, and the end point coordinates are changed from (40, 40) to (19, 44). The path diagram of the test process is shown in Figure 11, which shows that the USV successfully reaches the end point. During the test process, the PPO algorithm will explore the unknown action space to find a better strategy. When the episode reaches 77, the average reward value decreases sharply, and the number of steps is increased. At the 100th episode, the USV uses the least number of steps to reach the end point and receives the highest average reward. After changing the end point of the simulation environment to more complex areas, the models trained by the proposed algorithm can also guide USV to reach the end point. It indicates that the proposed algorithm has a strong generalization capability.

Figure 12 shows the average reward of the proposed algorithm with sensing capability after changing the end point of the simulation environment. The total training session is 100 episodes. As shown in Figure 12, the blue line takes a few time steps to reach the average reward convergence value, and the average reward for testing is stable at −233.84. This means that the USV can find a safe and collision-free path in the simulation environment after changing the end point of the path planning.

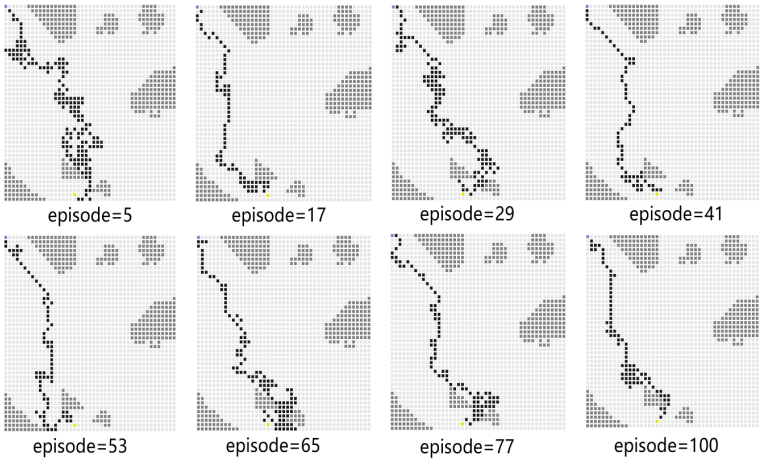


Figure 11. The path diagram on the testing map for Experiment 2.

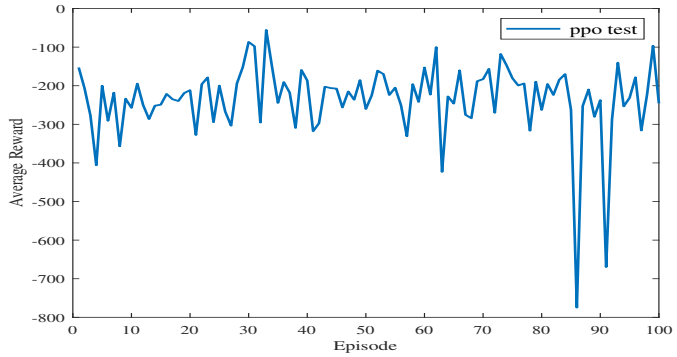


Figure 12. The average reward during testing in Experiment 2.

In Experiment 3, the generalization of the proposed algorithm is further verified by modifying the simulation map. In Experiment 3, the training map is changed to a new one, including the obstacles on the map and the end point coordinates. The path planning diagram for the test is shown in Figure 13. The algorithm proposed in this paper can make the USV bypass from above or below the obstacle and successfully reach the end point. At the 77th episode, the USV takes the least number of steps to reach the end point and receives the highest average reward. As shown in Figure 14, the blue line represents the

average reward convergence diagram after 100 episodes of testing. The average reward is −323.18.

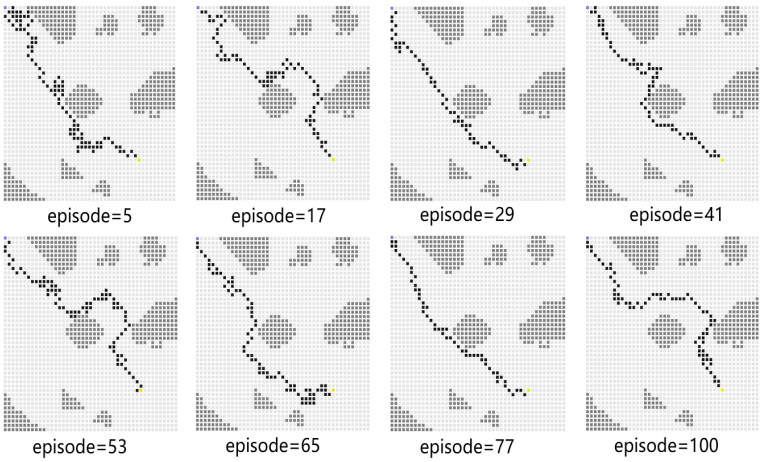


Figure 13. The path diagram on the testing map for Experiment 3.

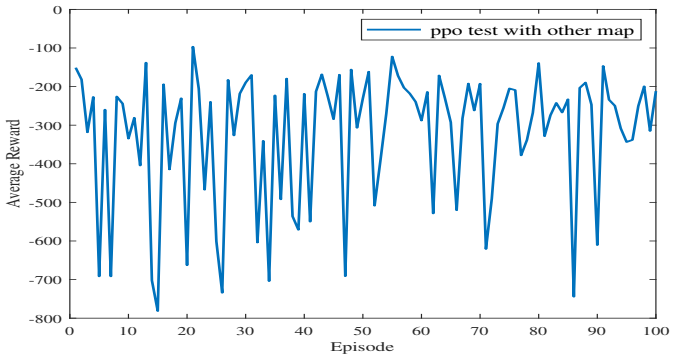
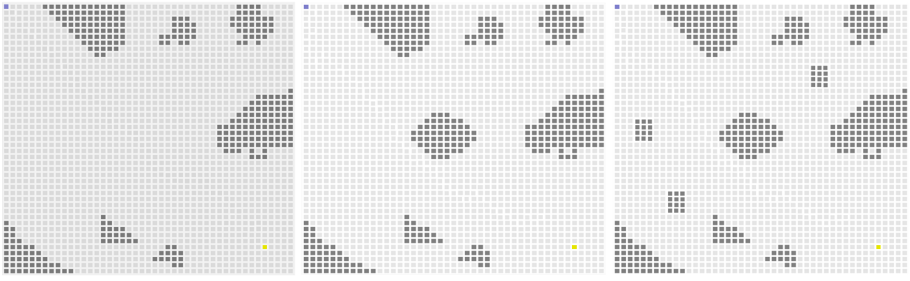


Figure 14. The average reward during testing in Experiment 3.

In Experiment 4, we increase the size of the training map sets, and parts of the maps are shown in Figure 15. As shown in Figure 16, the blue line represents the average reward convergence curve obtained by training with the three maps. The final average reward fluctuates between −100.28 and −168.54, indicating that the proposed algorithm basically tends to converge.



(a) Training map 1 (b) Training map 2 (c) Training map 3

Figure 15. Illustration of expanded training sets for generalization.

Figure 17 shows the path diagram in testing after expanding the training set. In some of the rounds shown therein, the USV successfully reaches the end point, which demonstrates the generalization capability of the algorithm proposed in this paper.

As shown in Figure 18, the blue line represents the average reward convergence diagram after 100 rounds of testing. The average reward is −186.84. After changing the training atlas, the average reward convergence curve obtained is shown in Figure 19, where

the solid black line represents the average reward convergence graph after training on a static graph, and the red dotted line represents the average reward convergence graph after training with three static maps.

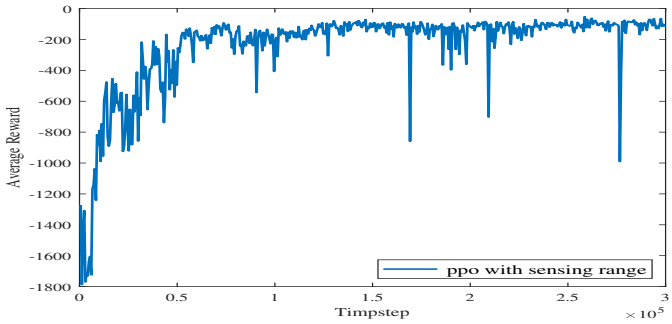


Figure 16. The average reward in Experiment 4.

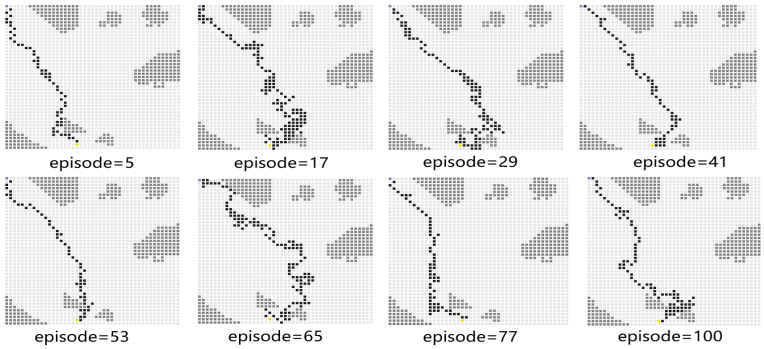


Figure 17. The path snapshots obtained during testing in Experiment 4.

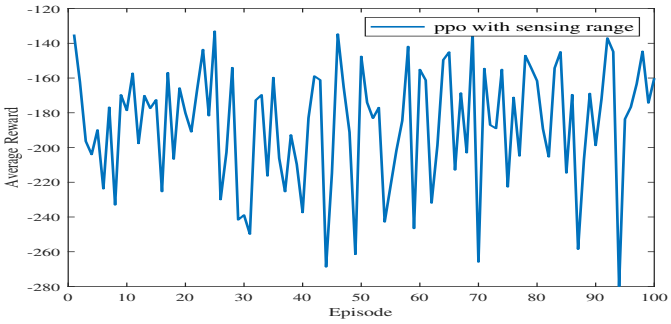


Figure 18. The average rewards during testing in Experiment 4.

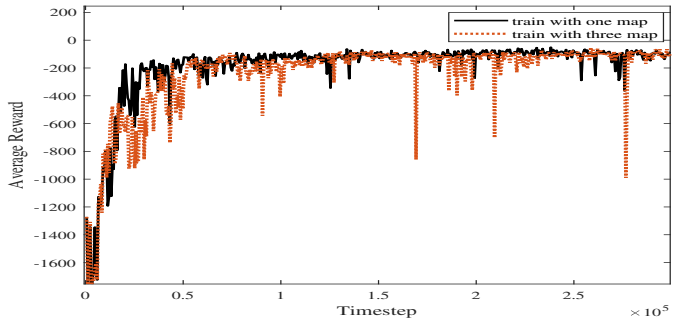


Figure 19. Comparison of average reward curves in Experiments 1 and 4.

4.2.3. Comparative Experiment

Experiment 5 verifies the effectiveness of the proposed algorithm by comparing its performance with several baseline algorithms. For comparison, the USV performs obstacle

avoidance tasks based on the SAC algorithm, the PPO algorithm, the DQN algorithm, and the proposed PPO algorithm with a sensing range. We compare the average reward obtained by these algorithms in the same scenario and the convergence time steps taken to reach the target position. The snapshots of different algorithm results are given in Figures 20–22. In the process of training, with the improvement of strategy, the USV becomes more and more certain and the average reward value is convergent. As shown in Figure 20, when the episode reaches 1958, the PPO algorithm explores a new action space, and the number of steps is increased. In Figure 21, at the 1634th episode, the USV falls into a local optimum near the end point, and the number of steps is increased. As shown in Figure 22, the number of steps for the USV to reach the end of each round gradually decreases, and the final average reward is convergent.

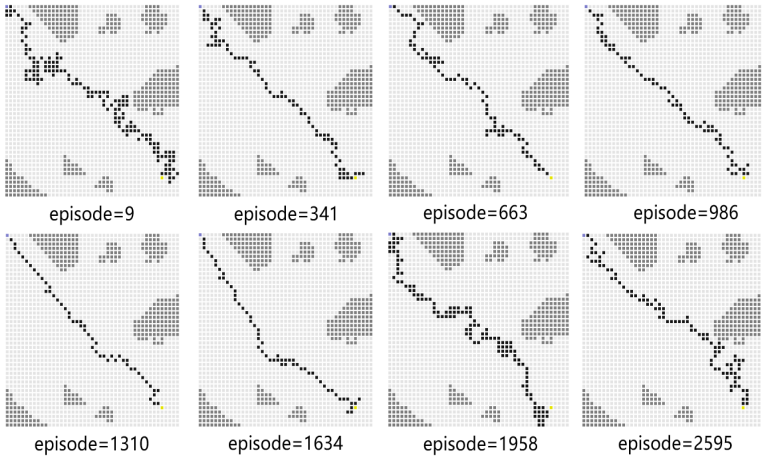


Figure 20. PPO-based path planning diagram.

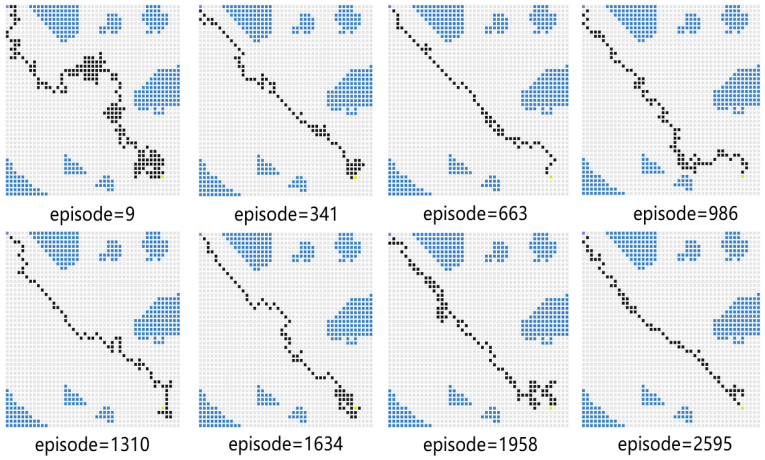


Figure 21. Path diagram with SAC.

Figure 23 shows the average reward curve of the four algorithms as the time steps increase in a static environment. The algorithm proposed in this paper has less fluctuation in the early training process than the PPO, DQN, and SAC algorithms. As shown in Table 3, the proposed algorithm, PPO algorithm, DQN algorithm, and SAC algorithm converge in 20,400, 59,400, 136,800 and 142,000 time steps, respectively. According to Figure 23 , SAC fluctuates greatly before the average reward converges. The proposed algorithm takes advantage of the improved perceptual capability of PPO and accumulates higher rewards. In the late stages of training, the average reward converges to around −117.574.

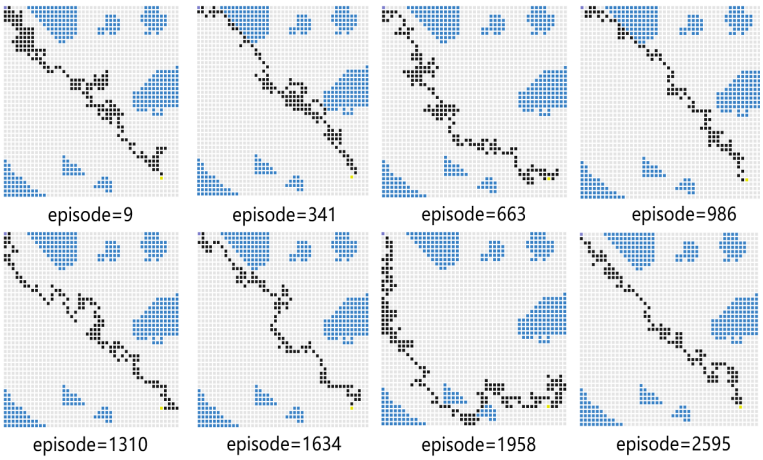


Figure 22. Path planning diagram with DQN.

Table 3. Comparison of experimental data in a static environment.

Comparative Experiment	Average Reward	Convergence Time Steps (10^4)
PPO with sensing range	−117.574	2.04
PPO	−535.18	5.94
DQN	−705.675	13.68
SAC	−840.58	14.2

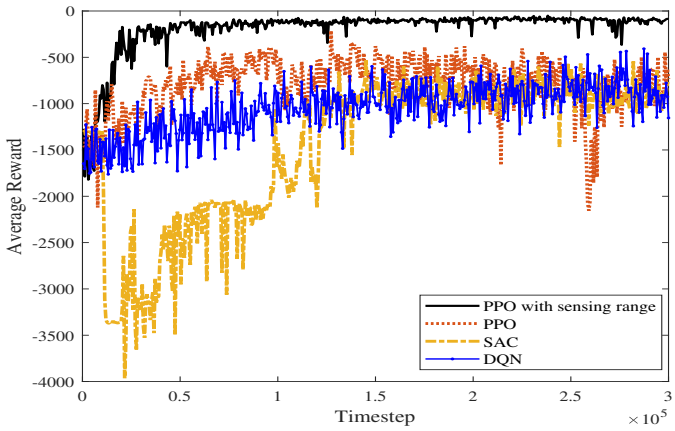


Figure 23. Average reward comparison graph of four algorithms in a static scene.

5. Conclusions and Future Work

In this paper, the path planning algorithm based on improved PPO for USVs is proposed. The convolutional layers are adapted as the state-feature extraction module for handling the sensing data in the sea environment. Such design helps the USV in making proper decisions when facing unknown environments, since the obstacle patterns have been learned with the CNN layers. Simulation experiments show that the average reward and convergence rate of the proposed algorithm are significantly better than the other baselines algorithms, which lays a foundation for subsequent research.

In our future work, given that a 2D raster model has a limitation in reflecting the influence of waves and winds at sea, we will expand our sea model to a 3D ocean area. In particular, we will focus on the stochastic influence of the environment on the USV strategies, especially in terms of motion control of the USV.

Author Contributions: Conceptualization, C.Y. and W.W.; Software, P.S. and X.Z.; Writing—original draft, P.S.; Writing—review and editing, P.S., W.W. and C.Y.; Supervision, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Nature Science Foundation under grant 62063011, the Yunnan Major Scientific and Technological Projects under grant 202202AG050002, the Open Foundation of Key Laboratory in Software Engineering of Yunnan Province under grant no. 2020SE502, the Yunnan Fundamental Research Projects under grant 202301AT070401, and partially supported by the Scientific Research Project of Yunnan Provincial Department of Education under grant 2023Y0418.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data generated or analyzed during the study are included in this published article.

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

References

1. Kurowski, M.; Thal, J.; Damerius, R.; Korte, H.; Jeinsch, T. Automated Survey in Very Shallow Water using an Unmanned Surface Vehicle. *IFAC Pap. Online* **2019**, *52*, 146–151. [CrossRef]
2. Jin, J.; Zhang, J.; Shao, F.; Lyu, Z.; Wang, D. A novel ocean bathymetry technology based on an unmanned surface vehicle. *Acta Oceanol. Sin.* **2018**, *37*, 99–106. [CrossRef]
3. Schofield, R.T.; Wilde, G.A.; Murphy, R.R. Potential field implementation for move-to-victim behavior for a lifeguard assistant unmanned surface vehicle. In Proceedings of the 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Philadelphia, PA, USA, 6–8 August 2018; pp. 1–2.
4. Liu, X.; Li, Y.; Zhang, J.; Zheng, J.; Yang, C. Self-adaptive dynamic obstacle avoidance and path planning for USV under complex maritime environment. *IEEE Access* **2019**, *7*, 114945–114954. [CrossRef]
5. Huang, Y.; Chen, L.; Chen, P.; Negenborn, R.R. Ship collision avoidance methods: State-of-the-art. *Saf. Sci.* **2020**, *121*, 451–473. [CrossRef]
6. Patle, B.K.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
7. Wang, D.; Zhang, J.; Jin, J.; Mao, X. Local collision avoidance algorithm for a unmanned surface vehicle based on steering maneuver considering colregs. *IEEE Access* **2021**, *9*, 49233–49248. [CrossRef]
8. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.A.; Burgard, W. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Cambridge, MA, USA, 2005.
9. Iijima, Y.; Hagiwara, H.; Kasai, H. Results of collision avoidance manoeuvre experiments using a knowledge-based autonomous piloting system. *J. Navig.* **1991**, *44*, 194–204. [CrossRef]
10. Churkin, V.I.; Zhukov, Y.I. Procedures for ship collision avoidance. In Proceedings of the IEEE Oceanic Engineering Society. OCEANS'98. Conference Proceedings (Cat. No. 98CH36259), Nice, France, 28 September–1 October 1998; Volume 2, pp. 857–860.
11. Hwang, C.N. The integrated design of fuzzy collision-avoidance and H[infty infinity]-autopilots on ships. *J. Navig.* **2002**, *55*, 117–136. [CrossRef]
12. Chang, K.Y.; Jan, G.E.; Parberry, I. A method for searching optimal routes with collision avoidance on raster charts. *J. Navig.* **2003**, *56*, 371–384. [CrossRef]
13. Szlapczynski, R. A new method of ship routing on raster grids, with turn penalties and collision avoidance. *J. Navig.* **2005**, *59*, 27–42. [CrossRef]
14. Niu, H.; Savvaris, A.; Tsourdos, A.; Ji, Z. Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles. *J. Navig.* **2019**, *72*, 850–874. [CrossRef]
15. Nie, Z.; Zhao, H. Research on robot path planning based on Dijkstra and Ant colony optimization. In Proceedings of the 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Shanghai, China, 21–24 November 2019; pp. 222–226.
16. Kuwata, Y.; Wolf, M.T.; Zarzhitsky, D.; Huntsberger, T.L. Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE J. Ocean. Eng.* **2013**, *39*, 110–119. [CrossRef]
17. Yao, P.; Zhao, R.; Zhu, Q. A hierarchical architecture using biased min-consensus for USV path planning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9518–9527. [CrossRef]
18. Wu, J.; Xue, Y.; Qiu, E. Research on Unmanned Surface Vehicle Path Planning Based on Improved Intelligent Water Drops Algorithm. In Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China, 6–8 November 2020; pp. 637–641.

19. Wei, A.; Yue, L.; Yanfeng, W.; Yong, H.; Guoqing, C.; Genwang, H. Design and Research of Intelligent Navigation System for Unmanned Surface Vehicle. In Proceedings of the 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China, 27–28 November 2020; pp. 1102–1107.
20. Woo, J.; Kim, N. Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. *Ocean. Eng.* **2020**, *199*, 107001. [CrossRef]
21. Zhang, X.; Wang, C.; Liu, Y.; Chen, X. Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning. *Sensors* **2019**, *19*, 4055. [CrossRef] [PubMed]
22. Jaradat, M.A.K.; Al-Rousan, M.; Quadan, L. Reinforcement based mobile robot navigation in dynamic environment. *Robot. Comput. Integr. Manuf.* **2011**, *27*, 135–149. [CrossRef]
23. Guan, W.; Cui, Z.; Zhang, X. Intelligent Smart Marine Autonomous Surface Ship Decision System Based on Improved PPO Algorithm. *Sensors* **2022**, *22*, 5732. [CrossRef]
24. Guo, S.; Zhang, X.; Du, Y.; Zheng, Y.; Cao, Z. Path planning of coastal ships based on optimized DQN reward function. *J. Mar. Sci. Eng.* **2021**, *9*, 210. [CrossRef]
25. Prianto, E.; Kim, M.; Park, J.H.; Bae, J.H.; Kim, J.S. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. *Sensors* **2020**, *20*, 5911. [CrossRef]
26. Habib, G.; Qureshi, S. Optimization and acceleration of convolutional neural networks: A survey. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 4244–4268. [CrossRef]
27. Lebedev, V.; Lempitsky, V. Speeding-up convolutional neural networks: A survey. *Bull. Pol. Acad. Sci. Tech. Sci.* **2018**, *66*, 799–811.
28. Krichen, M. Convolutional neural networks: A survey. *Computers* **2023**, *12*, 151. [CrossRef]
29. Tang, P.; Zhang, R.; Liu, D.; Huang, L.; Liu, G.; Deng, T. Local reactive obstacle avoidance approach for high-speed unmanned surface vehicle. *Ocean. Eng.* **2015**, *106*, 128–140. [CrossRef]
30. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
31. Kirk, R.; Zhang, A.; Grefenstette, E.; Rocktäschel, T. A survey of generalisation in deep reinforcement learning. *arXiv* **2021**, arXiv:2111.09794.

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