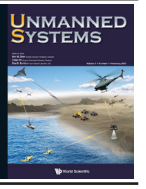
Unmanned Systems, Vol. 14, No. 2 (2026) 505–522

DOI: 10.1142/S2301385026500135

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us



An Intelligent Collision Avoidance Algorithm for Unmanned Surface Vehicles Based on Deep Reinforcement Learning

Bowen Zu, Xiang Wang, Xuehua Zhou, Zhiguo Zhou*

School of Integrated Circuits and Electronics,
Beijing Institute of Technology, Beijing 100081, China

To address the obstacle avoidance challenge for unmanned surface vehicles, this paper presents a novel intelligent algorithm based on deep reinforcement learning. The algorithm incorporates human demonstration experience data for quick convergence and efficient decision-making. It features an end-to-end framework for multi-sensor data processing and immediate action decisions. Both simulation and deployment experiments evidence the superiority of this algorithm.

Keywords: Unmanned surface vehicles; obstacle avoidance; deep reinforcement learning; artificial intelligence.

US

1. Introduction

1.1. *Background*

An unmanned surface vehicle (USV) is an autonomous system that operates on the surface of water, either semi-autonomously or fully autonomously. It can be equipped with various modules and sensors to meet the requirements of different tasks. Key advantages of USVs include high maneuverability, lightweight design and low energy consumption. They have extensive applications in both military and civilian domains [1, 2]. Research on USVs spans several technological fields, including motion control, environmental perception, autonomous navigation and positioning, and intelligent obstacle avoidance decision-making. Autonomous navigation and obstacle avoidance are critical aspects of USV intelligence, serving as primary indicators of their cognitive capabilities [3, 4].

Obstacle avoidance can be categorized into two main types: global planning and local planning [5, 6]. Global planning involves calculating an optimal path from an initial position to a target destination based on pre-existing environmental data. This approach relies on prior knowledge of environmental maps and is particularly suited for large-scale static environments. Local planning, on the other hand, focuses on generating a temporary, customized route to avoid obstacles and reach a designated target while adapting to immediate environmental changes. This method relies on real-time sensor data, enabling rapid responses to dynamic surroundings. This paper specifically examines local planning for obstacle avoidance in USVs.

Received 9 October 2024; Accepted 4 February 2025; Published 24 June 2025. This paper was recommended for publication in its revised form by editorial board member, Christos Verginis.
Email Addresses: *3220221583@bit.edu.cn; zhiguozhou@bit.edu.cn
*Corresponding author.

1.2. *Related work*

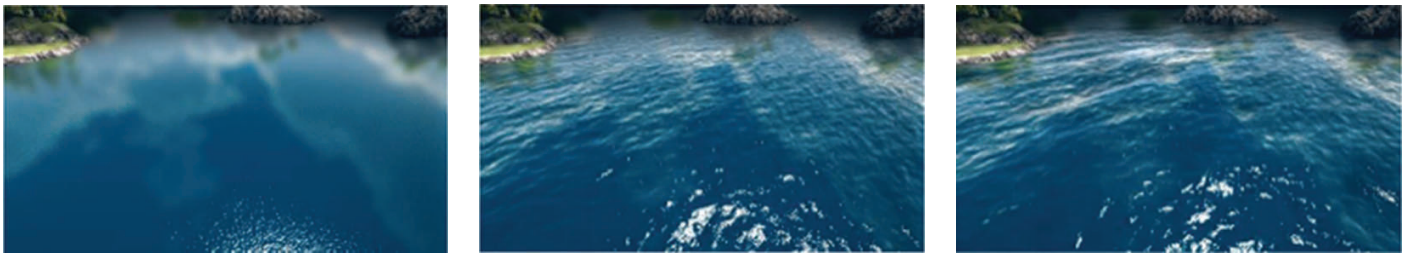
Most traditional path planning algorithms fall into three main categories: sampling-based, graph search-based and local-planning algorithms [7–10]. Sampling-based algorithms, such as the RRT algorithm, represent one prominent category. Chiang and Tapia [11] proposed an improved RRT algorithm that integrates COLREGS, achieving a higher success rate than obstacle avoidance algorithms based on model predictive control (MPC) combined with the artificial potential field (APF). However, the simulation does not account for the effects of winds, waves and currents. Wen *et al.* [12] developed an enhanced RRT algorithm that uses elliptic equations to restrict the sampling domain and plans real-time paths based on safe distances. Classical graph search-based algorithms include Dijkstra algorithm and the

A* algorithm. Fusic *et al.* [13] proposed an improved version of the Dijkstra shortest path algorithm, which reduces redundant node calculations and demonstrates the effectiveness of the algorithm on the V-REP testbed. He *et al.* [14] introduced a dynamic path planning algorithm based on A* and COLREGS rules, capable of effectively avoiding dynamic obstacles with known motion trajectories. However, this algorithm does not consider vehicle control or motion models. The above methods rely on sufficient prior information, such as environmental maps or obstacle data, which poses certain inherent limitations. Notable representatives of local planning algorithms include the APF method and the dynamic window approach (DWA). Lyu and Yin [15] proposed a "path-guided artificial potential field method", which considers COLREGS rules and the navigational characteristics of vehicles. Missura and Bennewitz [16] enhanced the conventional DWA algorithm by incorporating a "dynamic collision model" to predict potential future collisions, taking obstacle motion into account. Yuan *et al.* [17] proposed an improved DWA algorithm that adheres to COLREGS standards and considers the influence of environmental factors, such as wind and waves. However, both the APF method and the DWA algorithm are prone to local optimality in complex scenarios.

To address the limitations of conventional motion planning algorithms, various intelligent path planning algorithms have been explored [18–20]. Xu *et al.* [21] proposed a dynamic localization framework that leverages asynchronous LiDAR-camera fusion, optimized it through an attention mechanism, and designed a dual-servo rotary platform to accurately track UAVs, compensating for the limited detection range of sensors. This method provides a novel approach to multi-sensor fusion localization and navigation challenges in unmanned systems, with experimental results demonstrating its effectiveness in heterogeneous agent systems. Chen *et al.* [22] introduced an adaptive impedance control strategy aimed at achieving precise and flexible robot control. This was accomplished by developing an adaptive compensation controller and performing a system stability analysis, offering a robust foundation for the stable operation of unmanned systems. Deng *et al.* [23] presented a LiDAR-based framework for UAV detection, localization and tracking, targeting the challenge of accurate positioning in perceptually degraded environments. Their approach improves localization accuracy for heterogeneous agents in the absence of GNSS signals.

The DRL algorithm is an advanced machine learning approach widely applied in decision-making and control for unmanned systems. Wang *et al.* [24] proposed an optimal tracking control method based on the Actor–Critic framework, which effectively accounts for unknown dynamics, input nonlinearities and uncertainty disturbances, demonstrating superior accuracy and robustness. Wu *et al.* [25] improved the Dueling DQN algorithm for USVs by addressing state and action spaces to enable dynamic obstacle avoidance in simulation environments. Xu *et al.* [26] incorporated COLREGS into USV behavior regulations, leveraging enhanced DQN networks to ensure that trained obstacle avoidance decisions adhere to real-world navigation rules. Guo *et al.* [27] developed a DDPG-based autonomous navigation and obstacle avoidance model to handle continuous maneuvering control, translating navigation rules and encounters into navigational exclusion zones to enhance model training effectiveness. Wang *et al.* [28] introduced a comprehensive autopilot framework integrating path planning and tracking. This framework utilizes an elite replicating genetic algorithm (EGA) to generate optimal sparse waypoints, employs the B-spline technique to create smooth paths, and applies force-level strategy guidance using a depth deterministic strategy gradient (DDPG) model. Designed for underactuated, dynamic and spatially constrained conditions typical of USVs, this framework supports the deployment of autopilot systems in restricted waters. Hao *et al.* [29] proposed an autonomous obstacle avoidance algorithm based on an enhanced PPO, which incorporates a GRU network to improve learning efficiency, though the simulation experiments did not fully consider environmental influences. Cui *et al.* [30] combined global path planning and local obstacle avoidance by developing a navigation algorithm based on A* and SAC, addressing decision-making in uncertain and dynamic environments. However, the algorithm does not account for wind and wave effects, and obstacles are modeled as simple cylinders. Fan *et al.* [31] proposed an enhanced DQN algorithm to improve learning efficiency and constructed a virtual training and testing environment using Unity. Chun *et al.* [32] presented a localized obstacle avoidance algorithm based on PPO, which generates avoidance paths through quantitative collision risk assessment. Simulation results showed that this algorithm outperforms the A* algorithm in terms of effectiveness and adaptability.

This paper presents an intelligent obstacle avoidance algorithm based on deep reinforcement learning (DRL) to address the obstacle avoidance problem for USVs. The algorithm integrates human demonstration data into the PPO framework, enabling faster model convergence and improved learning efficiency. It adopts an end-to-end architecture that combines multiple sensor data as input to directly generate decision-making actions, allowing the vehicle to avoid obstacles and reach its designated target. The structure of this paper is as follows. Section 2 describes the methodology used to construct the simulation environment and the motion model of the USV. Section 3 details the DRL-based obstacle avoidance algorithm, including the underlying principles, network architecture and key design



(a) (b) (c)

Fig. 1. Effect of water body with different parameters. (a) Level I. (b) Level II. (c) Level III.

elements. Section 4 reports on the training and testing experiments as well as the analysis of the results for the obstacle avoidance algorithm. Finally, Sec. 5 concludes the paper with a summary and review of the work presented.

2. Simulation Environment

The obstacle avoidance algorithm based on DRL relies on trial-and-error learning, which requires feedback from numerous interactions between the intelligent agent and the environment. The algorithmic model is then trained in a simulation environment, where state data are collected to optimize learning strategies. Additionally, algorithms must be tested and validated on simulation platforms before being deployed in real-world scenarios. This section introduces a navigation environment and a USV model developed using the Unity engine and validates the models through experimental verification.

2.1. *Surface scene*

The surface scene consists of terrain and water bodies, which serve as platforms for training and testing autonomous obstacle avoidance capabilities. The specific methodology is as follows:

The terrain is constructed based on electronic maps, which are used as references to delineate water features. Virtual terrain generation is conducted automatically at a specified scale using geographic extent, elevation data, terrain characteristics and details of the selected area. A rendering tool modifies or adds geographic features, such as islands and reefs, to the global terrain, optimizing the scene resolution to a maximum of 10 cm. Water serves as a fundamental medium for navigation, and the simulation of surface effects directly influences training effectiveness. Simplified grid calculations are employed to simulate buoyancy forces, and the environment allows parameter adjustments to achieve various effects, including wave direction, intensity, current speed and simulated light interactions, such as reflection, refraction and scattering on the water surface. Figure 1 demonstrates the appearance of the water body under different parameter settings.

2.2. *USV model*

To accurately simulate the motion of the USV, it is essential to account for both its three-dimensional geometric characteristics and the attitude variations induced by external forces. In this study, geometric and physical models are developed based on the laboratory USV100.

2.2.1. *Grid model*

The USV100 measures 100 cm in length, 45 cm in width and 28 cm in height, with a weight of 15 kg. It is equipped with a twin-jet pump drive located at the stern, enabling a maximum speed of 1.5 mph. The geometric model is developed using 3ds-Max and subsequently imported into the simulation engine for further analysis. Tools are utilized to refine textures, mapping and colors, and to generate a mesh that accurately conforms to the surface of the hull. The 3D model and the corresponding mesh are shown in Fig. 2.

The grid model is influenced by forces from the water body, primarily buoyancy and wave forces [33]. The buoyancy force is calculated as shown in Eq. (1) and illustrated in Fig. 3(a).

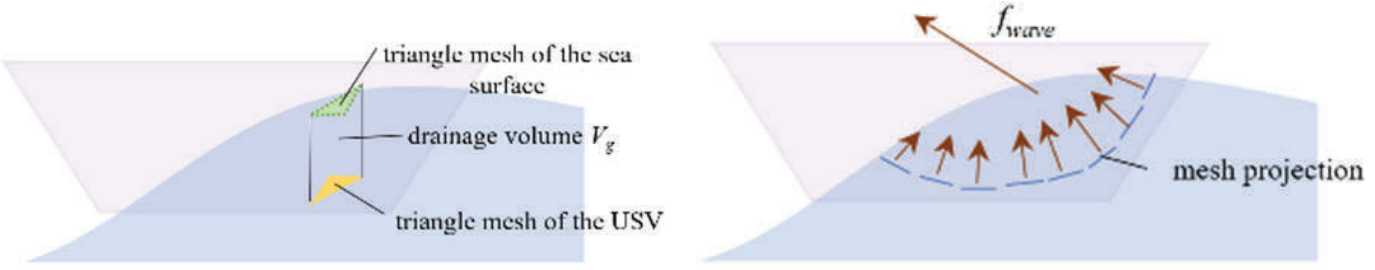
$$\mathbf{f}_{\text{buoy}} = \sum_{i=0}^{n} \begin{cases} -\rho \mathbf{g} V_g^i & \text{if mesh is underwater,} \\ 0 & \text{else,} \end{cases} \tag{1}$$

where ρ represents the water density, $\mathbf{g}$ denotes the gravitational vector, n indicates the number of triangular meshes on the hull and V_g^i refers to the volume enclosed by the ith triangular hull mesh and the water mesh.

The wave force calculation process is described in Eq. (2) and illustrated in Fig. 3(b).

(a) (b) (c)

Fig. 2. USV100 model. (a) Actual model. (b) Virtual model. (c) Grid model.



(a) (b)

Fig. 3. Schematic diagram of force on grid model. (a) Buoyancy. (b) Wave force.

$$\mathbf{f}_{\text{wave}} = \sum_{i=0}^{n} \begin{cases} -C_\omega k^i_{dot} \cdot \mathbf{n}^i_{\text{usv}} & \\ \|\mathbf{v}^i_{\text{wave}}\|_2 S^i_{\text{area}} & \text{if mesh is underwater,} \\ 0 & \text{else,} \end{cases} \tag{2}$$

where C_ω represents the force coefficient, $\mathbf{n}^i_{\text{usv}}$ denotes the normal vector of the ith hull triangle mesh, and $\mathbf{n}^i_{\text{ocean}}$ refers to the normal vector of the water mesh located above the ith hull triangle mesh. k^i_{dot} is the dot product of these two normal vectors. $\mathbf{v}^i_{\text{wave}}$ indicates the wave velocity above the ith hull triangle mesh, while S^i_{area} represents the area of the ith hull triangle mesh.

2.2.2. *Motion model*

In most cases, motion modeling is conducted based on three degrees of freedom. As shown in Fig. 4, x and y denote the positional coordinates; ψ represents the heading angle; u is the surge velocity; v is the sway velocity; and r is the yaw velocity.

In this study, a simplified MMG model [34, 35] is employed to evaluate the effects of multiple forces and moments on the motion of the USV. The MMG motion model is presented in Eq. (3).

$$\begin{cases} \dot{u} = \dfrac{f_u + (m + m_v)vr}{m + m_u} = \dfrac{f_{up} + f_{uw} + (m + m_v)vr}{m + m_u} \\ \dot{v} = \dfrac{f_v - (m + m_u)ur}{m + m_v} = \dfrac{f_{vp} + f_{vw} - (m + m_u)ur}{m + m_v} \\ \dot{r} = \dfrac{N}{I_z + J_z} = \dfrac{N_p + N_w}{I_z + J_z} \end{cases} \tag{3}$$

In this context, $\dot{u}$, $\dot{v}$ and $\dot{r}$ denote the time derivatives of the velocities. m, m_u and m_v represent the mass and the added mass in the u and v directions, respectively. f_u and f_v indicate the surge and sway forces acting on the USV. N denotes the yaw moment at the bow. Subscripts p and w represent the propeller thrust and the wave force, respectively. I_z and J_z indicate the rotational inertia and the added rotational inertia.

2.3. *Verification experiment*

This paper performs an emergency braking experiment and a spinning circle experiment to verify the accuracy of the simulation environment and the motion model. The emergency braking experiment evaluates the effects of varying drag forces on the vehicle in the simulation environment.

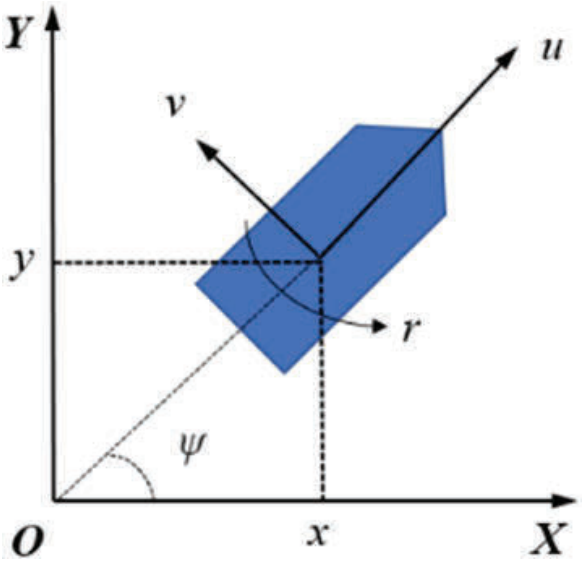


Fig. 4. The 3-DOF coordinate system of the USV.

The spinning circle experiment examines the consistency between the motion model and the actual motion behavior. The results of these experiments confirm that the motion error remains within an acceptable range, supporting the validity of the subsequent algorithm model deployment.

2.3.1. *Emergency braking experiment*

To evaluate the performance of emergency braking in both virtual and real environments, braking tests were conducted at identical velocities. The glide distance and stopping time were measured during the tests. The braking distance data under various velocities are summarized in Table 1, while Fig. 5 illustrates the braking trajectories at corresponding velocities in the virtual and real environments.

The experimental results indicate that the braking glide distance increases with rising velocity. Moreover, the glide distance in the simulation training environment is marginally greater than that in real-world conditions. This

Table 1. Emergency braking experimental data of USV100.

Initial velocity (m/s)	Actual sailing distance (m)	Visual sailing distance (m)	Deviation (m)	Relative error
0.3	2.26	2.38	0.12	5.31%
0.6	3.35	3.53	0.18	5.37%
0.9	4.43	4.68	0.25	5.61%
1.2	5.71	6.05	0.34	5.95%

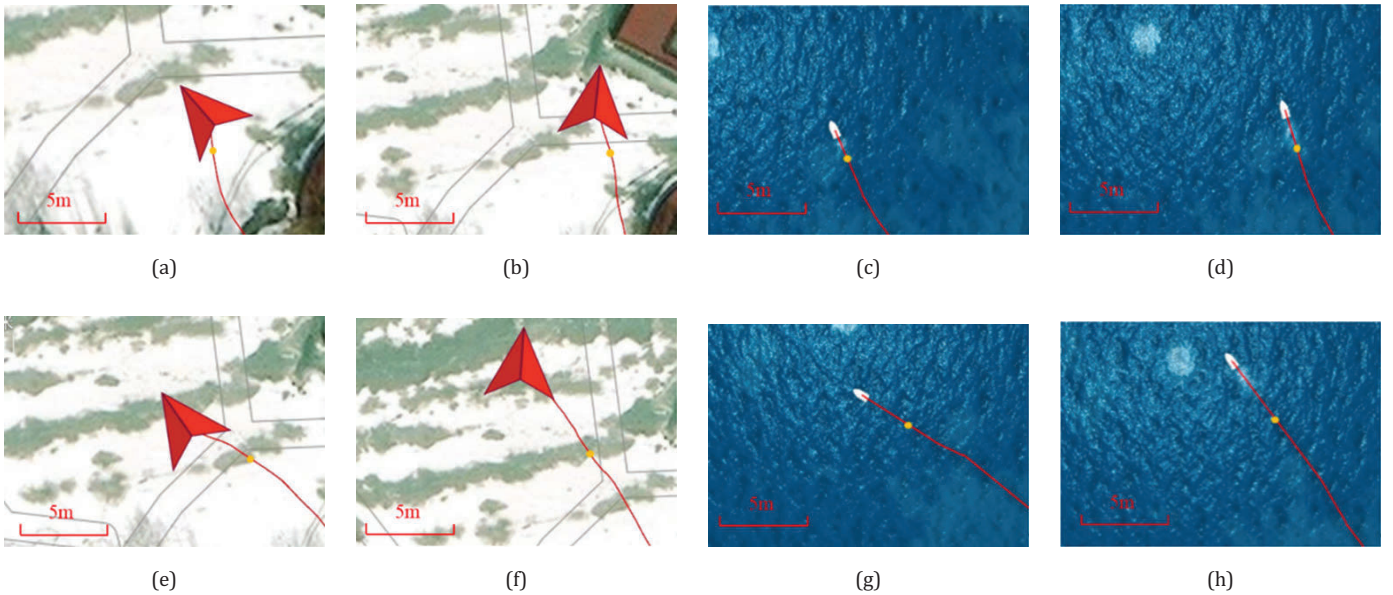


Fig. 5. Emergency braking track diagram of actual and virtual environments. (a) 0.3 m/s (actual). (b) 0.6 m/s (actual). (c) 0.3 m/s (virtual). (d) 0.6 m/s (virtual). (e) 0.9 m/s (actual). (f) 1.2 m/s (actual).

difference can be explained by the omission of environmental factors, such as air resistance, in this study. The overall results maintain a relative error within 6%, meeting the specified deviation requirements.

2.3.2. *Spin experiment*

The spin test is a widely used method for evaluating kinematic models in the marine field. In this method, the USV is required to maintain a constant straight-line velocity while steering to a prescribed rudder angle at the maximum achievable steering rate. Subsequently, the advance distance A_d, traverse distance T_r, initial turning diameter D_T and steady turning diameter D are recorded. The experiment compares data obtained in both virtual and actual environments at angular velocities of 0.35, 0.52 and 0.70 corresponding to scenarios 1, 2 and 3, respectively. Table 2 and Fig. 6 present detailed trajectory data and graphical representations.

A comparative analysis of the index data indicates that the deviations in inlet distance, cross-distance, initial spinning diameter, and spinning diameter remain below 8%. Moreover, the simulation environment accurately reproduces rotational motion, and the applied motion model complies with the vehicle maneuverability standards.

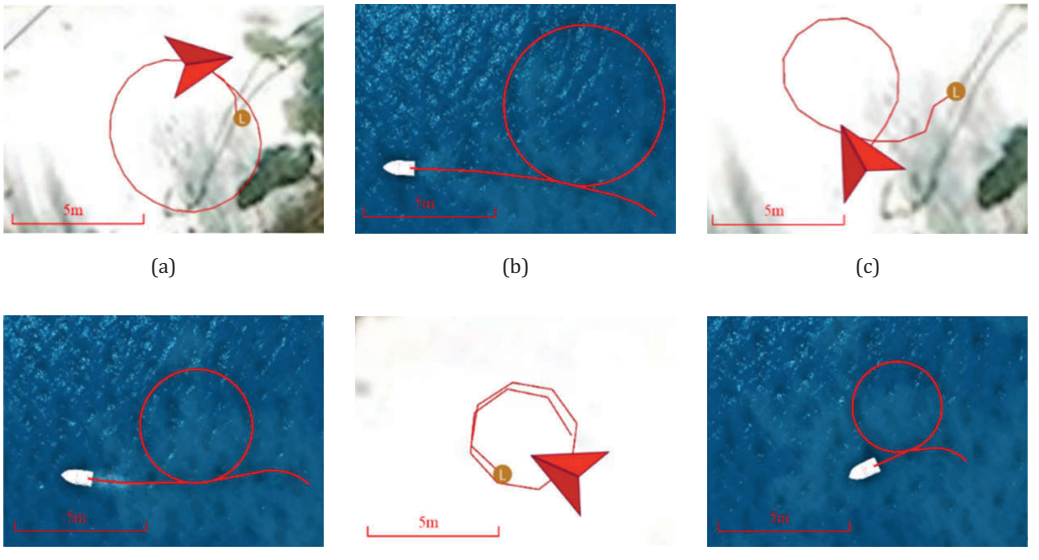
3. Collision Avoidance Algorithm Based on DRL

3.1. *Principles of deep reinforcement learning*

Reinforcement learning (RL) is a methodology positioned between supervised and unsupervised learning, emphasizing interactive and goal-oriented learning more strongly

Table 2. Spin experiment data of USV100.

Parameters of rotary ring	Leading distance (m)	Rotary initial diameter (m)	Horizontal distance (m)	Rotary diameter (m)
Actual 1	3.76	5.68	2.91	5.47
Virtual 1	3.88	5.94	3.12	5.58
Deviation 1	3.2%	4.6%	7.2%	2.0%
Actual 2	3.08	4.87	2.48	4.78
Virtual 2	3.26	5.04	2.59	4.93
Deviation 2	5.8%	3.5%	4.2%	3.2%
Actual 3	2.69	3.58	1.77	3.55
Virtual 3	2.83	3.72	1.92	3.81
Deviation 3	5.0%	3.9%	7.8%	6.8%



(a) (b) (c)

(d) (e) (f)

Fig. 6. Spin trajectories in virtual and actual environments of USV100. (a) Spin Circle 1 (actual). (b) Spin Circle 1 (virtual). (c) Spin Circle 2 (actual). (d) Spin Circle 2 (virtual). (e) Spin Circle 3 (actual). (f) Spin Circle 3 (virtual).

than other machine learning approaches. In RL, the agent is an intelligent entity capable of taking actions or making decisions within a given environment. Markov decision processes (MDPs) serve as the mathematical foundation and modeling framework for RL [36]. An MDP is typically defined by a quintuple (S, A, R, P, γ), comprising a state space, an action space, a reward function, a state transition function and a discount factor. In this context, S denotes the state space, which encapsulates the current environment and provides the basis for decision-making; A represents the action space, defined as the set of decisions the intelligent system can take in response to the current state; P refers to the state transition function, which is stochastic in nature, with the randomness stemming from the external environment.

$$p_t(s' \mid s, a) = P(S'_{t+1} = s' \mid S_t = s, A_t = a). \tag{4}$$

As illustrated in Eq. (4), the agent performs action a in the current state s, transitioning the system to state s'. R represents the reward provided by the environment to the agent upon completing the action. For long-term tasks, attention is typically given to the accumulated discounted rewards, where future rewards diminish progressively based on the discount factor γ. The definition of accumulated discounted reward is presented in Eq. (5).

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot r_{t+3} + \cdots = \sum_{k=0}^{T} \gamma r_{t+k}. \tag{5}$$

RL seeks to determine an optimal policy function that maximizes the expected accumulated discounted reward. Two key functions for evaluating and improving strategies are the state value function and the action value function, as defined in Eqs. (6) and (7), respectively.

$$V_\pi(s_t) = E_\pi[G_t \mid s_t] = E_\pi\left[\sum_{k=0}^{T} \gamma r_{t+k} \,\middle|\, s_t\right], \tag{6}$$

$$Q_\pi(s_t, a_t) = E_\pi[G_t \mid s_t, a_t] = E_\pi\left[\sum_{k=0}^{T} \gamma r_{t+k} \,\middle|\, s_t, a_t\right]. \tag{7}$$

The value function predicts the expected future reward based on the current state or sequence of actions. Accordingly, the optimal policy and the optimal value function are interdependent.

3.2. *Enhanced PPO algorithm*

DRL utilizes convolutional neural networks to process perceptual information in accordance with RL principles, establishing an end-to-end perception and control framework [37]. Over nearly a decade of development, several effective algorithms have emerged, among which the Actor–Critic framework is the most commonly employed. This framework involves two neural networks: Actor and Critic. The Actor is responsible for decision-making, while the Critic evaluates the strategy's effectiveness based on the value function. Representative algorithms include deep deterministic policy gradient (DDPG) [38], proximal policy optimization (PPO) [39], asynchronous advantage Actor–Critic (A3C) [40] and soft Actor–Critic (SAC) [41]. DDPG can be viewed as a combination of the PG algorithm and deep neural networks, serving as an extension of DQN in continuous action spaces. As a result, DDPG also suffers from Q-value overestimation and requires significant parameter tuning, making convergence challenging. A3C, on the other hand, employs the advantage function instead of the value function for action evaluation and integrates parallel training into A2C to enhance efficiency. However, this approach demands considerable parallel execution resources and encounters stability issues during training. In contrast, the PPO algorithm is an evolution of the PG algorithm, which simplifies the policy update mechanism, reduces algorithmic complexity, and improves learning efficiency. PPO-Clip, a variant of the PPO algorithm, employs a clipping technique to limit the discrepancy between old and new policies, ensuring stable policy updates. Consequently, this study adopts PPO-Clip as the primary algorithm. Despite the ability of PPO-Clip to achieve stable policy updates, it exhibits slow convergence during training, particularly in complex scenarios. To address this limitation, this study proposes a human demonstration module to accelerate the convergence of the model. The block diagram of the enhanced algorithm is shown in Fig. 7.

The process consists of two distinct phases: the first phase involves collecting human demonstration data, while the second focuses on training network updates. During the data acquisition phase, a human operator interacts with the environment instead of an intelligent agent. In each step, the operator observes the state of the simulated environment, including the position and orientation of the USV, the position and motion of obstacles, and the location of the target point, and selects an appropriate action. The operator then presses the corresponding button to drive the USV. Each action lasts 0.1 s. Subsequently, the state s_t, action a_t and reward r_t of the current step are recorded. If the current round is complete, the environment is reset. This process is repeated until the predefined maximum number of steps is reached. Incomplete rounds are discarded, and data from complete rounds is saved to a JSON file for backup.

In the network updating phase, the agent gathers experience through interaction with the environment and updates the network parameters to enhance its performance. As illustrated in Fig. 7, the PPO-Clip algorithm

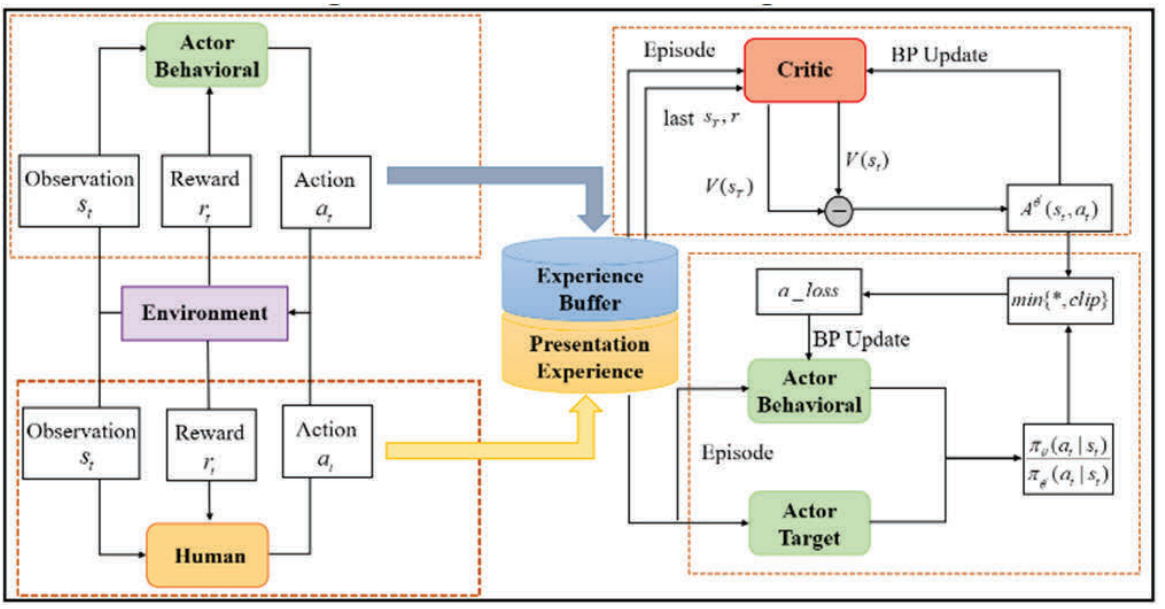


Fig. 7. Block diagram of the enhanced PPO algorithm structure.

includes two Actor networks: the behavioral network and the target network. The behavioral network interacts with the environment, collects interaction data τ and stores it in the experience pool.

$$\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T\}. \tag{8}$$

The target network refers to the network being optimized. During the update process, data must be randomly sampled from the experience pool. In traditional PPO-Clip, all data in the experience pool are generated by intelligent agents. In this study, prior to updating the network, human demonstration data is incorporated into the experience pool and mixed with the data collected by the intelligent agents to enhance their learning efficiency. PPO evaluates actions using the advantage function, as defined in the following equation:

$$\begin{aligned} A^{\theta'}(s_t, a_t) &= Q^{\theta'}(s_t, a_t) - V^{\theta'}(s_t) \\ &= \sum_{k=0}^{T-1} \gamma r_{t+k} + \gamma^T V^{\theta'}(s_{t+T}) - V^{\theta'}(s_t). \end{aligned} \tag{9}$$

The Critic network directly utilizes the advantage function, and optimization is performed via gradient descent. The Actor network, on the other hand, applies an importance weight, $r_t(\theta)$, to account for the discrepancy between the behavioral network and the target network. Finally, a clipping function is used to constrain the differences between the old and new policies within the range of $(1-\varepsilon, 1+\varepsilon)$, where a is a hyperparameter less than one. The overall objective function is defined in Eq. (11).

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)}, \tag{10}$$

$$\begin{aligned} L^{\mathrm{CLIP}}(\theta) = E_t[\min(r_t(\theta)A^{\theta'}(s_t, a_t), \mathrm{clip}(r_t(\theta), \\ \times 1-\varepsilon, 1+\varepsilon)A^{\theta'}(s_t, a_t))]. \end{aligned} \tag{11}$$

3.3. *Action space*

In this study, the action space is defined as a continuous space comprising two actions: propulsion and steering, as described in the following equation:

$$A = \{\delta_{\mathrm{push}}, \delta_{\mathrm{turn}}\}. \tag{12}$$

Here, δ_{push} represents the propulsion magnitude, scaled to the range [0, 1], indicating that only forward motion is permitted. δ_{turn} represents the steering magnitude, with a range of values from [−1, 1].

3.4. *State space*

State space refers to the collection of all possible states that the agent can occupy at a given point in time. In the context of obstacle avoidance, the USV determines the subsequent maneuver based on the prevailing operational state. The complexity of state space is a critical factor that influences the difficulty of RL problems. In this study, state space is

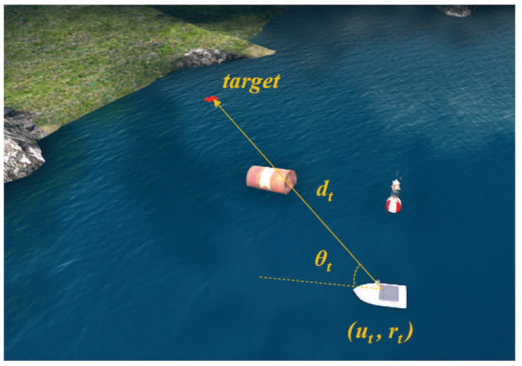
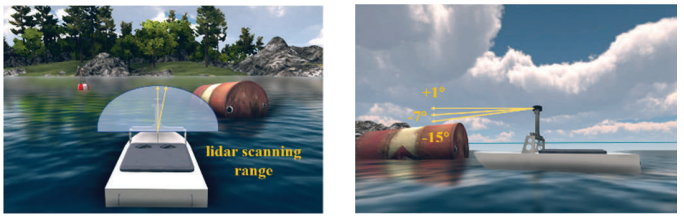


Fig. 8. Schematic diagram of the USV's self-status information.



(a) (b)

Fig. 9. Schematic of LiDAR information. (a) Front view. (b) Side view.

divided into two components: self-status information of the USV and lidar data.

(i) Self-status information

The self-status information of the USV consists of two elements: the current velocity (u_t, r_t) and the position relative to the target point (d_t, θ_t), as illustrated in Fig. 8.

In the illustration, d_t represents the Euclidean distance between the center of the USV and the target point at a given time t. θ_t denotes the azimuth angle of the USV relative to the target point, defined within the range $(-\pi, \pi)$. The linear and angular velocities of the USV at time t are represented by u_t and r_t, respectively. The self-status information, denoted as S_{usv} is expressed as a four-dimensional array $(d_t, \theta_t, u_t, r_t)$. Using this self-status information, the USV can determine its position relative to the target point and adjust its movement toward it.

(ii) LiDAR information

LiDAR demonstrates adaptability to varying light conditions in aquatic environments and performs robustly under such circumstances. In this study, the VLP-16, a 16-beam lidar from Velodyne, is utilized to simulate environment-aware detection. This device is capable of measuring both the distance and angle of obstacles. Due to the high dimensionality of point cloud data obtained from the 16-beam lidar, a down-sampling process is applied. The lidar data are shown in Fig. 9.

The process begins by converting the 3d point cloud into 2d laser scans. Angles of $+1°$, $-7°$ and $-15°$ are chosen to detect obstacles located directly in and below the front. For each laser scan, a portion of the $180°$ frontal range is selected and divided into equal intervals, with each interval covering $6°$. The rays within each interval are examined, and the shortest distance value is extracted as the state feature recorded by the lidar, which is then stored in the array *laser_data*. Consequently, the down-sampled lidar data are represented by a total of 90 features, denoted as S_{lidar}.

3.5. *Architecture of network*

As illustrated in Fig. 10, the left section illustrates the state information processing module. The self-status information of the USV is obtained using GPS and IMU, while lidar information is extracted by down-sampling the point cloud dataset. The combination of S_{usv} and S_{lidar} forms a complete

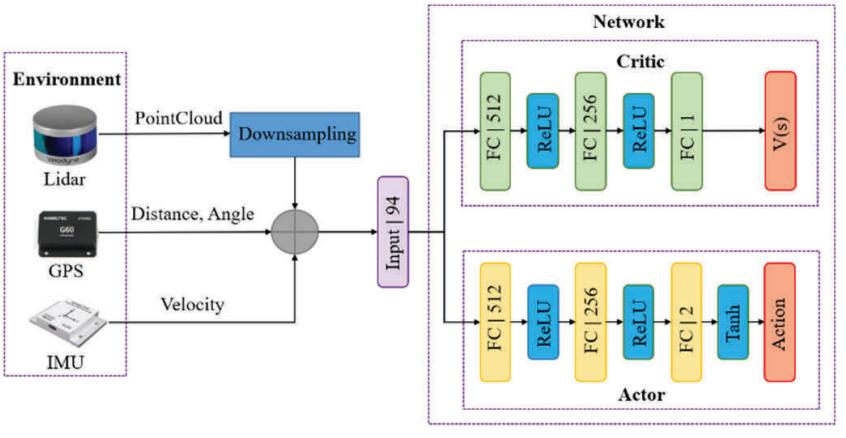


Fig. 10. Block diagram of the enhanced PPO algorithm structure.

state dataset with a total dimension of 94. The right section of the diagram depicts the network architecture. The Actor network consists of three fully connected layers activated by the ReLU function, with linear and angular velocities as outputs. The Critic network is composed of three fully connected layers, which jointly output the value assessment of the current state.

3.6. *Reward space*

The reward space is a critical component of the MDP, and it directly affects the efficiency of learning for the agent. The reward space designed in this study is described as follows:

(i) Goal reward

When the USV reaches the target point, the goal reward is granted, and the environment resets to initiate the next training episode. d_t represents the Euclidean distance between the center of the USV and the target point at a specific time step t. The pre-defined distance threshold λ has a default value of 1.0. If the value of d_t is smaller than λ, the USV is considered to have reached the target point.

$$r_{\text{goal}} = 100 \quad \text{if } d_t < \lambda. \tag{13}$$

(ii) Collision reward

A collision reward is given if the USV collides with an obstacle, and the environment resets afterward. The minimum value in the lidar information array at time point t is represented by $\min(laser\_data_t)$. A collision is deemed to have occurred when the value is less than the pre-established collision threshold η_c, which is set to a default value of 0.5.

$$r_{\text{collision}} = -100, \quad \text{if } \min(laser\_data_t) < \eta_c. \tag{14}$$

(iii) Obstacle reward

To reduce the likelihood of collisions and maintain the USV at the maximum possible distance from obstacles, obstacle avoidance thresholds η_d and η_u are defined. The severity of the penalty depends on the value of $\min(laser\_data_t)$ relative to η_d and η_u.

$$r_{\text{avoidance}} = \begin{cases} -(1/\min(laser\_data_t)) & \eta_d < \min(laser\_data_t) < \eta_u, \\ -1 & \min(laser\_data_t) < \eta_d, \\ 0 & \eta_u < \min(laser\_data_t). \end{cases} \tag{15}$$

(iv) Heading reward

The heading reward is positive when the angle between the heading of the USV and the line connecting the target point falls within a specified range, indicating that the USV is moving toward the target point.

$$r_{\text{heading}} \begin{cases} 0.5 & -1 < \theta < 1, \\ -0.5, & \text{else.} \end{cases} \tag{16}$$

(v) Distance reward

In addition to the heading direction, the distance between the USV and the target point serves as a critical factor in determining whether the USV is approaching the target point. This study uses the change in distance between the USV and the target point before and after an action is performed as the judgment basis. Cr is a constant with an initial value of -0.5.

$$r_{\text{distance}} = \text{Cr} * \delta_t. \tag{17}$$

(vi) Time reward

A fixed time penalty is applied at each time step to encourage the USV to reach the target point as quickly as possible.

$$r_{\text{time}} = -0.1. \tag{18}$$

In summary, the overall reward structure is described in the following equation:

$$R = \{r_{\text{goal}}, r_{\text{collision}}, r_{\text{avoidance}}, r_{\text{heading}}, r_{\text{distance}}, r_{\text{time}}\}. \tag{19}$$

The design of each reward weight follows these principles: initially, the allocation is determined based on task requirements and priorities. For example, reaching the target point, as the primary goal of the task, is assigned the highest weight, while collision, being the most undesirable outcome, is given the maximum negative weight. Subsequently, the weights of other rewards are adjusted based on experience and practical training outcomes. For instance, the initial value of the time penalty can be set as the maximum reward divided by the maximum number of steps per episode. As training progresses, the time penalty may become the primary penalty term, and its weight can be appropriately increased. Orientation rewards are useful in guiding the USV in the early training stages; however, assigning excessive weight to this factor can impede the development of obstacle avoidance skills.

4. Training and Testing

4.1. *Environment construction*

In this study, a virtual simulation training environment was developed using the Unity physics engine with a simulation step size of 0.02 s. The deep RL algorithm was implemented via the Pytorch training framework, offering a robust and efficient platform for training complex neural networks. The entire training workflow was executed using the OpenAI Gym framework. ROS and TCP/IP protocols were employed

to enable communication between the simulation environment and the algorithm. The training process was conducted on a high-performance computer equipped with an Intel(R) Core(TM) i7-10700 processor, a NVIDIA RTX 3070 graphics card and 16 GB of RAM. The primary software environments included Unity 2020.3.10f1c1, Ubuntu 20.04, ROS Noetic and Pytorch 1.13.0. The hyperparameters of the enhanced PPO algorithm were determined through historical analysis and extensive experimental validation, as presented in Table 3.

4.2. *Local collision avoidance training*

Local collision avoidance training utilizes a curriculum-based learning approach, where training progresses in stages with a gradual increase in environmental complexity. The initial training environment is configured as a 15-m by 15-m area containing five static buoy obstacles and one red target point. The wave condition is set to Level I, as shown in Fig. 11. At the start of each episode, the position of the target point is randomly updated while avoiding overlap with the positions of the obstacles, and the USV is initialized at a fixed position in the bottom-left corner of the scene. Notably, the target points and scene boundaries are virtual objects that function solely as observation aids without influencing the motion of the USV. Moreover, these virtual objects cannot be perceived by the USV.

The training process is conducted using three algorithms, including the one described in this paper, with two sensor configurations. The first configuration employs a single-line LiDAR, while the second uses a 16-line LiDAR, both subjected to the same down-sampling process. The number of training steps is set to 100,000, and the reward function curve is presented in Fig. 12.

As shown in Fig. 12, the overall curve of the algorithm proposed in this paper is positioned in the upper-left region for both configurations, indicating advantages in convergence speed and convergence level. During the early training phase, the reward values derived from the experience sequences generated by the interaction between the USV and the environment are relatively low, exhibiting performance inferior to that of human demonstration data. Incorporating human demonstration data accelerates learning, as reflected in the training curves by faster convergence and shorter convergence time. In the later stages of training, the human demonstration data provide multiple examples of obstacle avoidance, enhancing the instincts of the USV for decision-making, which is reflected in the curve as a higher cumulative reward value. Comparing Figs. 12(a) and 12(b), the training curves under the single-line LiDAR configuration exhibit significant fluctuations and fail to

Table 3. Principal hyperparameters of the enhanced PPO algorithm.

Parameters	Value	Definition
max_steps	300,000	Maximum number of steps in the training process
max_ep_steps	200	Maximum number of steps in a round
update_timestep	3000	Interval steps for updating policy network parameters
K_epochs	80	Number of turns included in a strategy update
gamma	0.99	Discount factor
eps_clip	0.2	Cutting factor
lr_actor	0.0003	Learning rate of Actor network
lr_critic	0.001	Learning rate of the Critic network
expl_noise	0.1	Variance of the noise

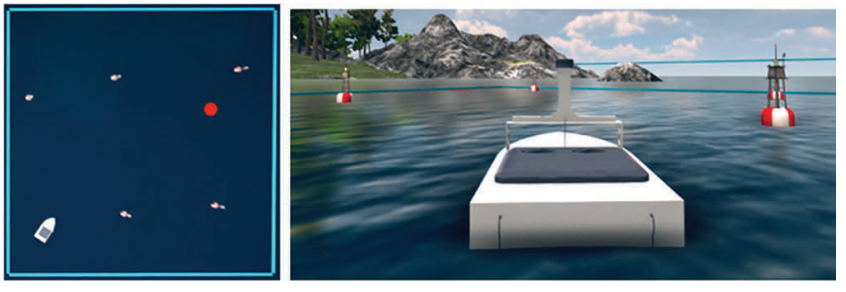
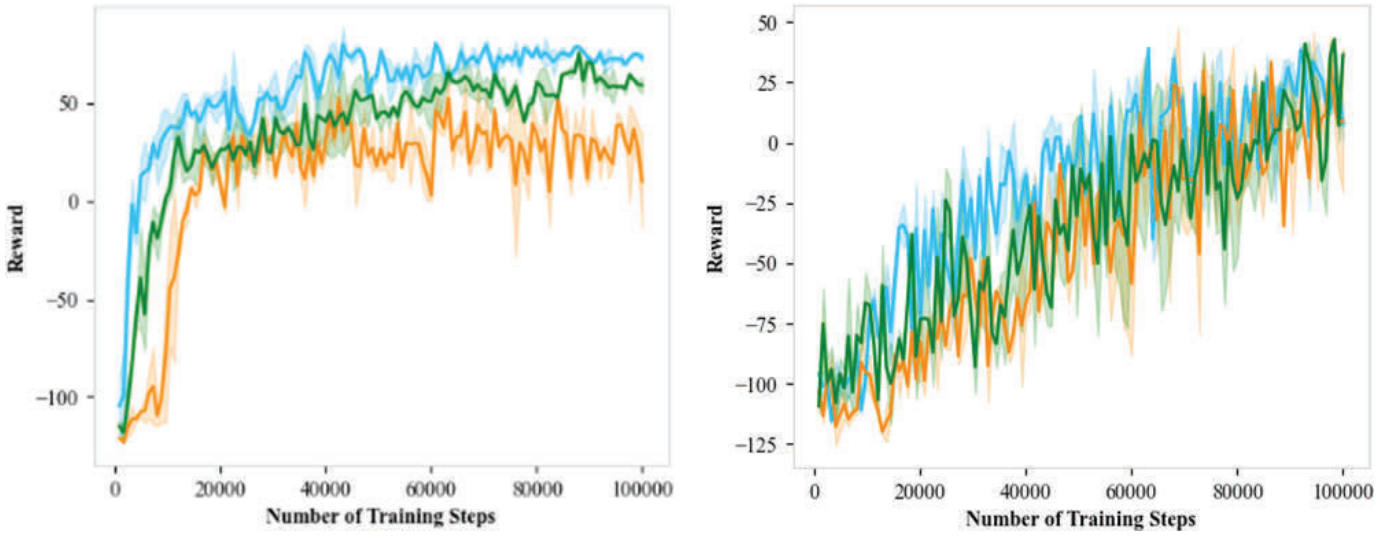


Fig. 11. Phase 1 training scenario.



(a) (b)

Fig. 12. Phase 1 training curve. (a) 16-Line lidar. (b) Single-line lidar.

converge completely. This can be attributed to the inability of the single-line LiDAR to detect low-lying reefs, resulting in a lower obstacle avoidance success rate.

The training environment for the second phase is set up in a 30-m by 30-m area, comprising five static buoy obstacles, two dynamic oil drum obstacles, one static reef obstacle and one red target point. The wave level is set to Level II, as shown in Fig. 13. Static obstacles remain fixed in position, while dynamic obstacles move slightly under the influence of wind and waves. At the beginning of each round, the positions of the USV and the target point are randomly generated, ensuring no overlap with the obstacles.

The model underwent additional training over 200,000 steps, utilizing the three algorithms in sequence. As shown in Fig. 14, the PPO and SAC curves exhibit significant fluctuations due to the increased complexity of the scene during the second stage, where the USV fails to develop an effective obstacle avoidance strategy. Additionally, the random initialization at the start of each iteration results in varying outcomes: in some cases, the USV successfully reaches its destination, yielding higher reward values, while in others, it fails to reach the destination, resulting in lower rewards. This leads to a pattern of fluctuations in the curves between higher and lower values. In contrast, the enhanced algorithm, incorporating human demonstration data, exhibits superior learning performance. Although minor fluctuations are observed in its curves, the overall trend is upward, eventually converging toward a stable equilibrium.

After the training process, the algorithm was tested and evaluated to assess its performance. The test environment introduced additional obstacles to those in the Phase 2 training environment and adjusted the wave level to Level 3. Each algorithm was tested 100 times, and the resulting data were analyzed across three metrics: success count, failure count and collision count. The results of this analysis are displayed in Fig. 15.

The success count represents the number of times the USV reaches the target point from the starting point within the maximum number of round steps. The collision count refers to the number of instances where the USV collided during navigation. The lost count denotes the number of

Fig. 13. Phase 2 training scenario.

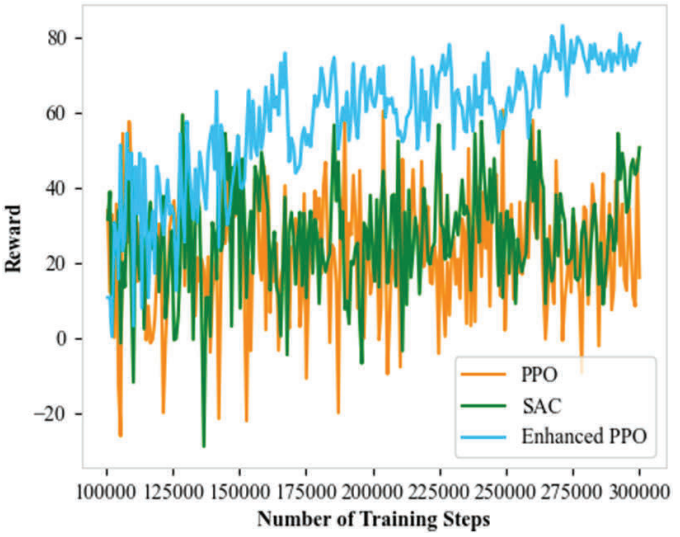
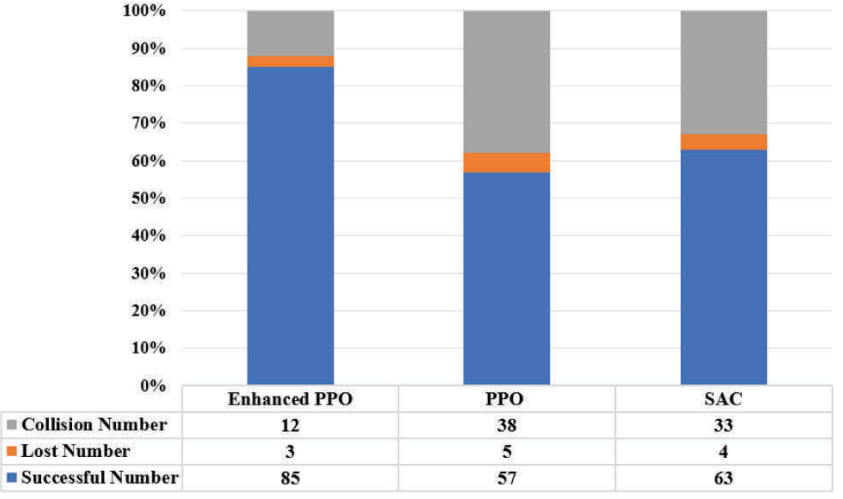


Fig. 14. Phase 2 training curve.



	Enhanced PPO	PPO	SAC
Collision Number	12	38	33
Lost Number	3	5	4
Successful Number	85	57	63

Fig. 15. Experimental test data of obstacle avoidance for each algorithm.

instances where the USV neither collided nor reached the target point within the maximum number of round steps. The trajectory for the test scenario is shown in Fig. 16.

As depicted in Fig. 16, the obstacle avoidance performance of each algorithm is correlated with the cumulative reward curve observed during training. Specifically, a higher reward value after stable convergence indicates more effective obstacle avoidance during testing.

To evaluate the model's robustness to noise, Gaussian noise of varying intensities is added to the original lidar data, and obstacle avoidance is tested in the same scenario. The laser point cloud, after incorporating noise, is visualized using the Rviz tool, as shown in Fig. 17.

The algorithm described in this paper was tested in the specified test environment. Each configuration was evaluated 100 times, and three metrics were recorded: the number of successes, the number of lost trips and the number of collisions, as shown in Fig. 18.

From the analysis of Fig. 18, it can be observed that the obstacle avoidance performance of the model is not significantly affected when the Gaussian noise variance does not exceed 0.5. However, when the noise variance increases to 1.0, the obstacle avoidance success rate decreases by approximately 10%. At higher noise levels, the shape of the point cloud generated by scanning obstacles becomes noticeably distorted, which interferes with the model's

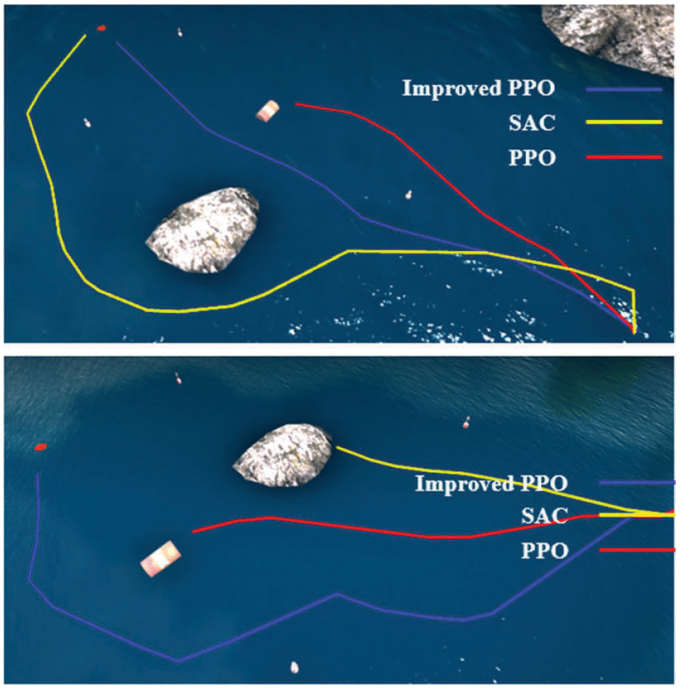
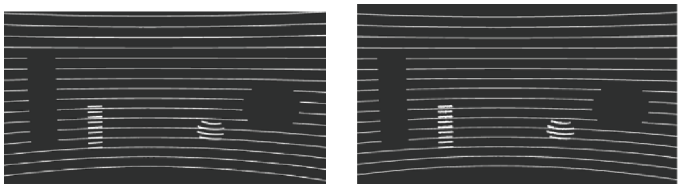


Fig. 16. Trajectory of the USV in the test scenario.



(a) (b)

Fig. 17. Lidar point cloud with different levels of noise added. (a) No noise added. (b) Adding Gaussian noise with a variance of 0.05. (c) Adding Gaussian noise with a variance of 0.5. (d) Adding Gaussian noise with a variance of 1.0.

decision-making process. Overall, the model demonstrates robustness against Gaussian noise interference.

4.3. *Large-scale scene navigation*

In scenarios with high state space complexity and significant distances between the robot and the target point, DRL algorithms often face challenges in achieving an optimal balance between exploration and exploitation. When applied across a variety of environments, the frequency of positive rewards tend to decrease as the size of the training area expands, which may hinder the convergence of the model. To address this, a global path is first generated using a global path planning algorithm, followed by the selection of local target points at fixed intervals via a tracking algorithm. This method effectively transforms the problem of large-scale navigation into a series of local obstacle avoidance tasks.

The test scenario consists of a square area measuring approximately 1000 m by 1000 m. As shown in Fig. 19(a), the global path is computed by inputting the raster map of the scene into the global path planning algorithm. Figure 19(b) depicts the local target points determined by the pure tracking algorithm, as well as the trajectory of the USV implemented with the local obstacle avoidance algorithm model. It can be observed that the USV sequentially navigates toward designated target points, making necessary adjustments to the trajectory and avoiding obstacles based on the prevailing conditions.

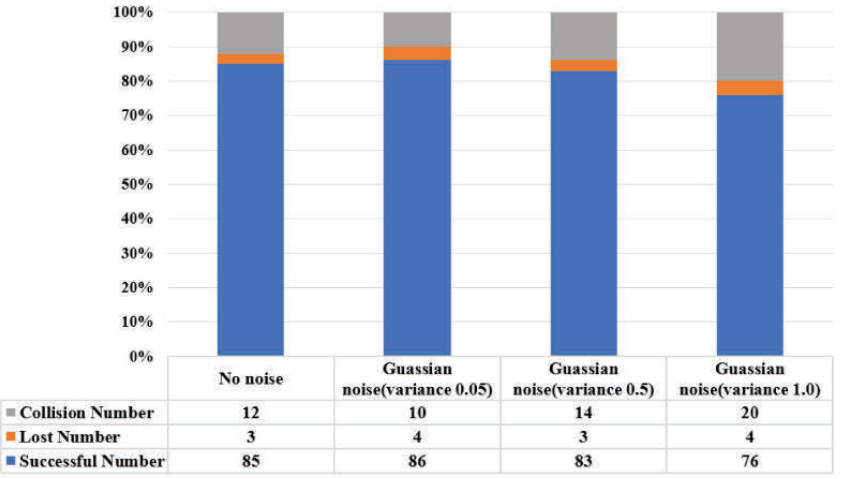
4.4. *Deployment verification*

Deployment verification was conducted using USV100, which is composed of two main components: ROS serving as the upper-layer application development framework for implementing algorithms and functional packages, and Pixhawk4 functioning as the lower-layer controller to execute algorithms for forward and reverse paddle control. Data exchange between these two components is facilitated by MAVROS. The configuration details of USV100 are provided in Table 4.

The deployment process is illustrated in Fig. 20. USV100 perceives its surroundings using onboard sensors, such as LiDAR and GPS. With the support of ROS, the collected information is published as topics. The algorithm node subscribes to these topics, processes the input data through a decision network for inference and generates control commands. These commands are transmitted via MAVROS to Pixhawk4, which interpret the signals and executes the corresponding drive controls. Simultaneously, the obstacle avoidance trajectory and various sensing data are transmitted to a shore-based monitoring computer via a 4G or Wi-Fi module.

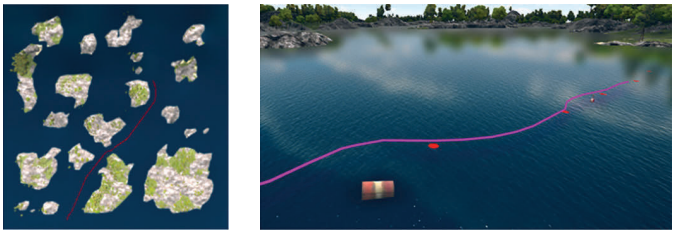
4.4.1. *System performance analysis*

The obstacle avoidance performance of USV100 is evaluated in this study in terms of computational cost and system latency. For the computational cost, the number of floating-point operations per second (FLOPS) is used as the metric for model computation. As described in Sec. 3.5, the PPO network comprises both ACTOR and CRITIC components, but only the ACTOR component is required for decision-making during model inference. The ACTOR network consists of three fully connected layers with a total of 180,482 parameters and 179,712 FLOPS. The model is deployed on a Jetson Nano board with an ARM A57 CPU



	No noise	Guassian noise(variance 0.05)	Guassian noise(variance 0.5)	Guassian noise(variance 1.0)
Collision Number	12	10	14	20
Lost Number	3	4	3	4
Successful Number	85	86	83	76

Fig. 18. Obstacle avoidance test data under different levels of noise.



(a) (b)

Fig. 19. Trajectory map of the USV for large-scale scene. (a) Global paths. (b) Local target points and local paths.

running at 1.43 GHz. The inference time is measured over 100 rounds, and the average inference time is determined to be 25 ms.

The system latency is broadly divided into three parts: sensing data frame update, inference computation and response delay. Time synchronization for multi-sensor data is performed using the time synchronizer mechanism provided by ROS. The lidar point cloud update frequency is 10 Hz, and the USV position update frequency is 50 Hz, resulting in a synchronized frequency of approximately 7.8 Hz. Consequently, the sensing data frame update delay is 128 ms. As noted earlier, the inference computation delay is

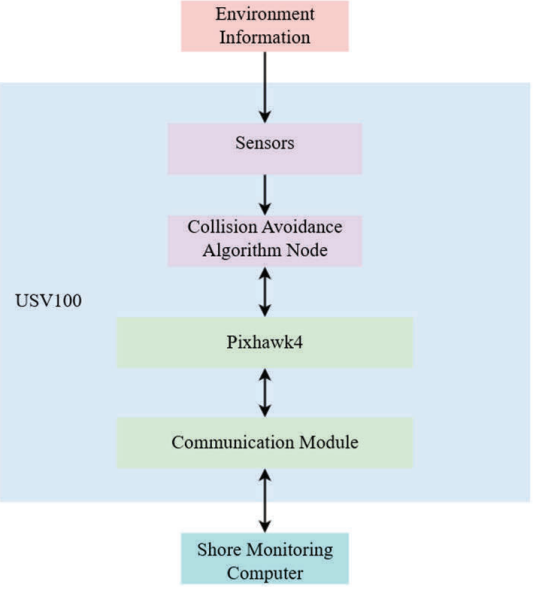


Fig. 20. Autonomous obstacle avoidance process for USV.

25 ms, while the response delay of the motor-propeller unit, treated as an inertial link, is measured to be 23 ms. The total system latency is therefore 176 ms, with the sensing data update delay being the primary contributor. This delay

Table 4. USV100 configuration.

Master controller	Jetson Nano B01
Laser radar	Velodyne16
Camera	Intel RealSense D455
Communication system	Communication system
Bottom controller	Pixhawk4

Table 5. Deployment test experiment results.

Algorithm	Success rate	Average length of time
DWA	70%	35 s
PPO	50%	28 s
SAC	60%	33 s
Enhanced PPO	80%	22 s

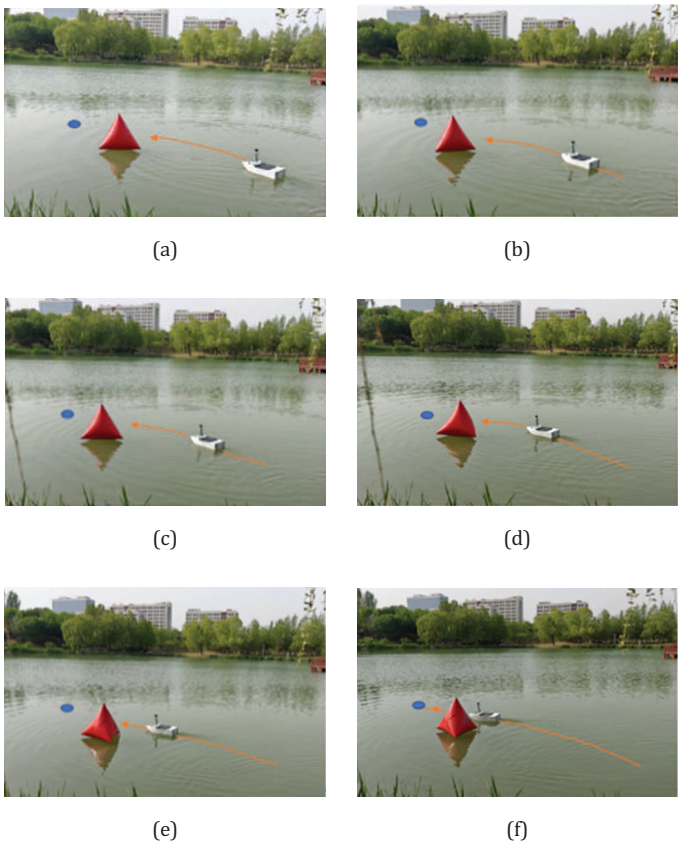


Fig. 21. USV100 obstacle avoidance scene. (a) t = 1 s. (b) t = 4 s. (c) t = 7 s. (d) t = 10 s. (e) t = 13 s. (f) t = 17 s.

can be reduced by increasing the radar point cloud update frequency. For the inference computation, acceleration can be achieved through model quantization and pruning, provided that the model's accuracy is maintained.

4.4.2. *Obstacle avoidance test*

To validate the effectiveness of the model, obstacle avoidance tests were conducted in real-world aquatic environments. The coordinate system used is a north-east-up coordinate system, where the positive x-axis points east, and the positive y-axis points north. The positioning of various obstacles reflect changes in the aquatic environment. Table 5 compares the obstacle avoidance success rates and average time for each algorithm under real-world conditions. Figure 21 illustrates the obstacle avoidance process of the algorithm proposed in this study.

The experimental results indicate that the obstacle avoidance performance of the algorithm in real-world environments is slightly inferior to that in the simulation environment. This discrepancy arises because the simulation environment cannot fully account for environmental factors and noise, and the real-world system inherently involves delays and sensor inaccuracies. Nevertheless, the experiments demonstrate that the proposed algorithm exhibits good generalization performance and can effectively apply the trained network to real-world aquatic scenarios.

5. Conclusion

This paper proposes an intelligent obstacle avoidance algorithm for USVs based on DRL. The algorithm employs an end-to-end framework that directly outputs decision actions by integrating multiple sensor data, thus simplifying the manual design of decision rules compared to traditional algorithms. Human demonstration data are incorporated into PPO to accelerate the convergence process and improve the capability of the model to avoid obstacles. To enhance the robustness and generalization of the model, a refined water surface environment and dynamics model is constructed to simulate the influence of water currents. The reliability of the constructed model is validated through comparative experiments. The proposed algorithm has been demonstrated to guide USVs effectively to avoid obstacles and reach designated positions in both simulated and real-world environments, proving its feasibility and practicality. Furthermore, this study explores the integration of the local obstacle avoidance algorithm with a global path planning algorithm through simulation experiments, demonstrating the potential of the algorithm for applications in large-scale scenarios.

Despite the strong performance of the proposed DRL algorithm in local obstacle avoidance for USVs, certain limitations remain. In this study, the effect of waves on USV motion is simulated using grid-based computation in Unity, which does not fully reflect the hydrodynamic characteristics of actual USVs. Additionally, environmental factors such as wind, complex currents and surface reflections are not considered. Future research will involve the use of professional system modeling and simulation tools, such as MWORKS to develop more accurate USV motion and environmental models, which will then be combined with Unity for co-simulation. Furthermore, the algorithm has been tested only in an island scenario; additional scenarios will be introduced to improve the robustness and generalization of the model.

ORCID

Bowen Zu https://orcid.org/0009-0009-7385-8263
Xiang Wang https://orcid.org/0009-0009-7255-6984
Xuehua Zhou https://orcid.org/0009-0003-5467-5038
Zhiguo Zhou https://orcid.org/0000-0003-4207-5759

References

[1] E. Ziajka-Poznańska and J. Montewka, Costs and benefits of autonomous shipping— A literature review, *Appl. Sci.* **11**(10) (2021) 4553.

[2] Z. Liu *et al.*, Unmanned surface vehicles: An overview of developments and challenges, *Annu. Rev. Control* **41** (2016) 71–93.

[3] S. Blindheim, T. A. Johansen and I. B. Utne, Risk-based supervisory control for autonomous ship navigation, *J. Mar. Sci. Technol.* **28**(3) (2023) 624–648.

[4] J. Liu *et al.*, Testing and evaluation for intelligent navigation of ships: Current status, possible solutions, and challenges, *Ocean Eng.* **295** (2024) 116969.

[5] C. Liu *et al.*, Human–machine cooperation research for navigation of maritime autonomous surface ships: A review and consideration, *Ocean Eng.* **246** (2022) 110555.

[6] J. Xia *et al.*, Research on collision avoidance algorithm of unmanned surface vehicle based on deep reinforcement learning, *IEEE Sens. J.* **23**(11) (2022) 11262–11273.

[7] J. Xie *et al.*, Reinforcement-learning-based asynchronous formation control scheme for multiple unmanned surface vehicles, *Appl. Sci.* **11**(2) (2021) 546.

[8] Z. Cui, W. Guan and X. Zhang, Collision avoidance decision-making strategy for multiple USVs based on Deep Reinforcement Learning algorithm, *Ocean Eng.* **308** (2024) 118323.

[9] G. Xia *et al.*, Unmanned surface vehicle collision avoidance trajectory planning in an uncertain environment, *IEEE Access* **8** (2020) 207844–207857.

[10] T. Inan and A. F. Baba, Building a hybrid algorithm based decision support system to prevent ship collisions, *J. Fac. Eng. Archit. Gazi Univ.* **35**(3) (2020) 1213–1230.

[11] H. T. L. Chiang and L. Tapia, COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation, *IEEE Robot. Autom. Lett.* **3**(3) (2018) 2024–2031.

[12] N. Wen, R. Zhang, J. Wu and G. Liu, Online planning for relative optimal and safe paths for USVs using a dual sampling domain reduction based RRT∗ method, *Int. J. Mach. Learn. Cybern.* **11**(12) (2020) 2665–2687.

[13] S. J. Fusic, P. Ramkumar and K. Hariharan, Path planning of robot using modified dijkstra Algorithm, in *Proc. 2018 National Power Engineering Conf. (NPEC)* (IEEE, 2018), pp. 1–5.

[14] Z. B. He, C. G. Liu and X. M. Chu, Dynamic anti-collision A-star algorithm for multi-ship encounter situations, *Appl. Ocean Res.* **118** (2022) 16.

[15] H. Lyu and Y. Yin, Fast path planning for autonomous ships in restricted waters, *Appl. Sci.* **8**(12) (2018) 2592.

[16] M. Missura and M. Bennewitz, Predictive collision avoidance for the dynamic window approach, in *Proc. Int. Conf. Robotics and Automation* (IEEE, 2019), pp. 8620–8626.

[17] X. Yuan *et al.*, Unmanned vessel collision avoidance algorithm by dynamic window approach based on COLREGs considering the effects of the wind and wave, *J. Mar. Sci. Eng.* **11**(9) (2023) 1831.

[18] M. Martelli *et al.*, An outlook on the future marine traffic management system for autonomous ships, *IEEE Access* **9** (2021) 157316–157328.

[19] R. Cimurs, J. H. Lee and I. H. Suh, Goal-oriented obstacle avoidance with deep reinforcement learning in continuous action space, *Electronics* **9**(3) (2020) 411.

[20] R. Zhang, X. Hongwei and Y. Kexin, Autonomous collision avoidance system in a multi-ship environment based on proximal policy optimization method, *Ocean Eng.* **272** (2023) 113779.

[21] Y. Xu *et al.*, LCDL: Towards dynamic localization for autonomous landing of unmanned aerial vehicle based on lidar–camera fusion, *IEEE Sens. J.* **24**(16) (2024) 26407–26415.

[22] Z. Chen *et al.*, Adaptive impedance control for docking robot via Stewart parallel mechanism, *ISA Trans.* **155** (2024) 361–372.

[23] C. Deng *et al.*, LiDAR depth cluster active detection and localization for a UAV with partial information loss in GNSS, *Unmanned Syst.* **13**(2) (2024) 491–503.

[24] N. Wang *et al.*, Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle, *IEEE Trans. Neural Netw. Learn. Syst.* **32**(7) (2020) 3034–3045.

[25] X. Wu *et al.*, The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method, *Knowl.-Based Syst.* **196** (2020) 105201.

[26] X. Xu *et al.*, Intelligent collision avoidance algorithms for USVs via deep reinforcement learning under COLREGs, *Ocean Eng.* **217** (2020) 107704.

[27] S. Guo *et al.*, An autonomous path planning model for unmanned ships based on deep reinforcement learning, *Sensors* **20**(2) (2020) 426.

[28] N. Wang *et al.*, Autonomous pilot of unmanned surface vehicles: Bridging path planning and tracking, *IEEE Trans. Veh. Technol.* **71**(3) (2021) 2358–2374.

[29] S. Hao *et al.*, USV collision avoidance decision-making based on the improved PPO algorithm in restricted waters, *J. Mar. Sci. Eng.* **12**(8) (2024) 1428.

[30] Z. Cui *et al.*, Autonomous navigation decision-making method for a smart marine surface vessel based on an improved soft actor–critic algorithm, *J. Mar. Sci. Eng.* **11**(8) (2023) 1554.

[31] Y. Fan, Z. Sun and G. Wang, A novel intelligent collision avoidance algorithm based on deep reinforcement learning approach for USV, *Ocean Eng.* **287** (2023) 115649.

[32] D. H. Chun, M. I. Roh and H. W. Lee, Deep reinforcement learning-based collision avoidance for an autonomous ship, *Ocean Eng.* **234** (2021) 109216.

[33] W. Gan *et al.*, Research on key technology of unmanned surface vehicle motion simulation based on Unity3D, in *Proc. OCEANS 2022* (IEEE, 2022), pp. 1–5.

[34] H. Yasukawa and Y. Yoshimura, Introduction of MMG standard method for ship maneuvering predictions, *J. Mar. Sci. Technol.* **20**(1) (2015) 37–52.

[35] J. Yan *et al.*, Identification modeling and prediction of ship maneuvering motion based on LSTM deep neural network, *J. Mar. Sci. Technol.* **27**(1) (2022) 125–137.

[36] T. Degris, P. M. Pilarski and R. S. Sutton, Model-free reinforcement learning with continuous action in practice, in *Proc. 2012 American Control Conf. (ACC)* (IEEE, 2012), pp. 2177–2182.

[37] V. Mnih *et al.*, Playing atari with deep reinforcement learning, preprint (2013), arXiv:1312.5602.

[38] T. P. Lillicrap *et al.*, Continuous control with deep reinforcement learning, preprint (2015), arXiv:1509.02971.

[39] J. Schulman *et al.*, Proximal policy optimization algorithms, preprint (2017), arXiv:1707.06347.

[40] V. Mnih *et al.*, Asynchronous methods for deep reinforcement learning, in *Proc. Int. Conf. Machine Learning* (PMLR, 2016), pp. 1928–1937.

[41] T. Haarnoja *et al.*, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *Proc. Int. Conf. Machine Learning* (PMLR, 2018), pp. 1861–1870.

Bowen Zu received his bachelor's degree in Electronic Information Engineering from Beijing Institute of Technology and continued to pursue a postgraduate degree. His main research interest includes intelligent vehicle perception and navigation and intelligent unmanned system.

Xiang Wang received his bachelor's degree in Measurement and Control Technology and Instrument from Wuhan University of Technology. In 2023, he entered the Beijing Institute of Technology to pursue a postgraduate degree. His main research interest includes embodied intelligent robot navigation.

Xuehua Zhou received his bachelor's degree in Biomedical Engineering from Beijing Institute of Technology, Beijing, China, in 2010, and his M.E. degree in Electronic Science and Technology from Beijing Institute of Technology, Beijing, China, in 2013. He is currently an Engineer at the School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing, China. His research interests include artificial intelligence, unmanned platform, real-time simulation and image processing.

Zhiguo Zhou received his B.E. degree in Measurement and Control Technology and Instrument from the Huazhong University of Science and Technology, Wuhan, Hubei, China, in 1998, and his M.E. and Ph.D. degrees in Communications and Information Systems from Beijing Institute of Technology, Beijing, China, in 2004 and 2009, respectively. He is currently an Associate Professor at the School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing, China. His research interests include artificial intelligence, unmanned platform, virtual reality, object detection and image processing.