Journal of Systems Engineering and Electronics
Vol. 34, No. 5, October 2023, pp.1343 – 1358

LSTM-DPPO based deep reinforcement learning controller for path following optimization of unmanned surface vehicle

XIA Jiawei[1,2], ZHU Xufang[3,*], LIU Zhong[1], and XIA Qingtao[1]

1. School of Weaponry Engineering, Naval University of Engineering, Wuhan 430033, China; 2. Qingdao Campus, Naval Aviation University, Qingdao 266041, China; 3. School of Electronic Engineering, Naval University of Engineering, Wuhan 430033, China

Abstract: To solve the path following control problem for unmanned surface vehicles (USVs), a control method based on deep reinforcement learning (DRL) with long short-term memory (LSTM) networks is proposed. A distributed proximal policy optimization (DPPO) algorithm, which is a modified actor-critic-based type of reinforcement learning algorithm, is adapted to improve the controller performance in repeated trials. The LSTM network structure is introduced to solve the strong temporal correlation USV control problem. In addition, a specially designed path dataset, including straight and curved paths, is established to simulate various sailing scenarios so that the reinforcement learning controller can obtain as much handling experience as possible. Extensive numerical simulation results demonstrate that the proposed method has better control performance under missions involving complex maneuvers than trained with limited scenarios and can potentially be applied in practice.

Keywords: unmanned surface vehicle (USV), deep reinforcement learning (DRL), path following, path dataset, proximal policy optimization, long short-term memory (LSTM).

DOI: 10.23919/JSEE.2023.000113

1. Introduction

As a small surface platform for operation, unmanned surface vehicles (USVs) offer several advantages such as low cost, high speed, intelligence, and no casualties. It has broad application prospects in military applications, scientific expeditions, commercial development, hydrographic surveys, search and rescue, relay communications, etc. [1]. Motion control is the core of the techniques used for USV autonomous navigation, which can help a USV to smoothly complete the tasks.

USV path following has always been a popular but difficult topic in the relevant research fields. It is defined as the process in which a USV is driven by a control system without considering temporal constraints to depart from any initial position and reach the expected path while eventually arriving at the designated destination along with the path [2]. A USV with no side thruster or perhaps with a side thruster fails during high-speed navigation, which is common in a typical underactuated system. This type of system is complex to some extent because the number of control inputs is lower than the number of degrees of freedom (DOF) to be controlled.

Many researchers have developed various control algorithms to address the path following needs of underactuated USV. The research methods for path following mainly include the Lyapunov direct method [3,4], feedback linearization [5], sliding mode control [6,7], the back-stepping method [8,9], active disturbance rejection control [10], robust control [11], and model predictive control [12,13]. These conventional control methods often need to consider the prior knowledge of an USV dynamic model and cope with strong coupling, complexity, and uncertainty of the system. Reinforcement learning (RL) is the opposite of the feedback evaluation of optimal control theory since it is mainly based on valuation and can operate when the system model is unknown [14]. Unlike optimal control theory, RL is based on evaluative, rather than being instructive; feedback comes in different forms, which may or may not include partial knowledge of the environment or the system. With the continuous progress in the field of artificial intelligence, RL has become an important branch of machine learning and made remarkable progress in the field of control and decision [15−18].

There have been plenty of attempts to implement deep RL (DRL) techniques in path following control for unmanned vehicles [19,20]. Among them, the conver-

Manuscript received August 18, 2021.
*Corresponding author.
This work was supported by the National Natural Science Foundation (61601491), the Natural Science Foundation of Hubei Province (2018CFC865), and the China Postdoctoral Science Foundation Funded Project (2016T45686).

gence performance of RL controller has received the most attention. Typically, path following controller is composed by a cascaded system of a guidance law and a low-level controller. One promising approach is to have DRL controller acting as both of them [21,22]. DRL network output the speed and steering command according to the USV's cross-track error and heading error relative to the desired path, as well as the kinematic parameters of the platform. Although the techniques to ensure the robustness to disturbances were adopted such as the Ornstein-Uhlenbeck process in control actions [23], the convergence of the DRL controller wasn 't strictly proved. The other approaches input DRL network with guidance law [24], such as line-of-sight (LOS) [25] and vector field guidance (VFG) [26], which do not impose complex proof of stability on the path following control. However, these studies were only verified in single cases or limited scenarios.

Although USV can effectively converge in the simulation with a DRL controller, as a black-box method, the lack of interpretability makes its security questioned. It is of vital importance to avoid catastrophic safety accidents in engineering practice. Hence, improving the security while ensuring convergence with the USV path following control is of great concern.

In order to bridge the aforementioned research gap, a path following method for underactuated USV with the feature of both convergency and security is proposed. Firstly, dynamics and kinematics models of USV are established, and then the LOS guidance law and USV motion states are combined as the DRL controller's input. Inspired by training deep learning networks with massive data sets, the concept of data sets for path following is established, which allows a distributed stochastic sampling of the problem's domain that enhances the generalization of the task at the cost of a larger training process. Furthermore, the long short-term memory-distributed proximal policy optimization (LSTM-DPPO) algorithm is designed to process large-scaling training work by distributed computing. In addition, considering that path following control is a strong time-relevant problem, the LSTM network layer is introduced in the neural network. The performance of the proposed method was evaluated using simulation experiments in various scenarios. The analysis results demonstrate that our method is effective for path following control of USVs.

Salient contributions in this paper can be summarized as follows.

(i) The concept of path dataset is proposed for the first time by generating a variety of possible path combinations to cover the USV sailing scenarios, which helps the DRL controller adaptable to the critical situation.

(ii) A distributed proximal policy optimization reinforcement learning method with the LSTM layer is designed to meet the requirements of extensive training and time-relevant control.

The rest of this paper is organized as follows. Section 2 presents the USV dynamics model, guidance method, and reward functions design. Section 3 introduces the path dataset and simulation environment. Section 4 gives theoretical background and technical implementation details of the LSTM-DPPO method. Section 5 discusses the validation and comparative analysis of the proposed method. Finally, Section 6 presents the main conclusions and potential for future work.

2. USV path following

Fig. 1 presents the USV path-following control process. The process can be regarded as the transfer of data between three modules, including the USV dynamic model, USV virtual environment, and controller. A USV dynamic model can use the control input to estimate the state at the next moment and send kinematics parameters to the virtual environment. Subsequently, the virtual environment determines the position of the USV on the map in a real-time manner and calculates the guidance parameters based on the mission path.

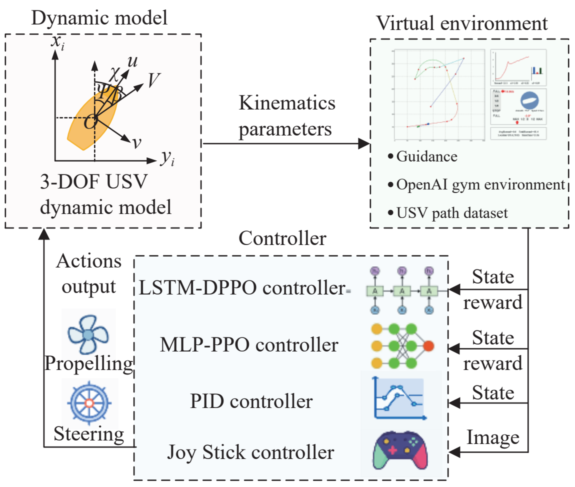


Fig. 1 Diagram of a USV path following control

Depending on the principles of feedback control used, controllers require different inputs. For instance, RL controllers such as LSTM-DPPO or multilayer perceptron version of proximal policy optimization (MLP-PPO) mainly require state and reward input, whereas conventional control methods like proportion-integral-derivative (PID) controllers only need state data. Thanks to the graphical simulation environment of OpenAI, a joystick controller can compete with an artificial intelligence (AI) algorithm. In this case, the controller only requires images as input, and a closed-loop of feedback exists. This process is similar to someone playing a video game.

The dynamic process of USV path following requires the use of a USV dynamic model to calculate the response under a given control. Meanwhile, a guidance system should be designed to determine the expected heading to evaluate whether the USV effectively follows the desired route. The rest of this section provides detailed information about the components of the USV path following system.

2.1 USV dynamic model

Path following by a USV focuses on motion in a two dimensional (2D) plane. Therefore, this paper ignores the concepts of heave, roll, and pitch but uses the 3-DOF kinematic model [25], which can not only address the problems of accuracy and network generalization during the path following in a plane but also meet the requirements for providing a convenient calculation method.

The coordinate system used in this work is shown in Fig. 2.

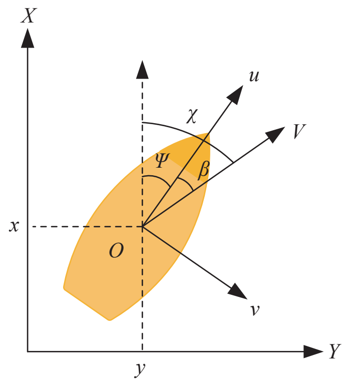


Fig. 2 Coordinate system of a USV

In Fig. 2, X and Y are the East and North direction; u and v stand for surge and sway; ψ, β, and χ denote heading angle, side slip angle, and course angle respectively; USV position can be described as $\boldsymbol{\eta} = [x, y, \psi]^{\mathrm{T}}$, and speed as $\boldsymbol{v} = [u, v, r]^{\mathrm{T}}$. The 3-DOF USV kinematic model can be described as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\boldsymbol{\eta})\boldsymbol{v} \tag{1}$$

where $\boldsymbol{R}(\boldsymbol{\eta})$ is a rotation matrix from the body-fixed frame to the Earth-fixed inertial frame.

$$\boldsymbol{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The horizontal planar dynamic motion equation of the USV can be expressed as

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v} = \boldsymbol{\tau}_c + \boldsymbol{\tau}_e \tag{3}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}(\boldsymbol{v})$ is the Coriolis and centripetal matrix, $\boldsymbol{D}(\boldsymbol{v})$ denotes the damping matrix. $\boldsymbol{\tau}_c = [\tau_u, 0, \tau_r]^{\mathrm{T}}$ denotes the control inputs, τ_u is the surge force, τ_r is the yaw moment. $\boldsymbol{\tau}_e = [\tau_{\mathrm{eu}}, \tau_{\mathrm{ev}}, \tau_{\mathrm{er}}]^{\mathrm{T}}$ is the environmental forces due to wind, waves, and ocean currents. τ_{eu}, τ_{ev}, τ_{er} denote the disturbances of the surge, sway, and yaw, respectively.

$$\begin{cases} \boldsymbol{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix} \\ \boldsymbol{C} = \begin{bmatrix} 0 & 0 & c_{13} \\ 0 & 0 & c_{23} \\ -c_{13} & -c_{23} & 0 \end{bmatrix} \\ \boldsymbol{D} = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix} \\ c_{13} = -m_{22}v - \dfrac{1}{2}(m_{23} + m_{32})r \\ c_{23} = m_{11}u \end{cases}$$

Equation (3) can be simplified as

$$\begin{cases} \dot{u} = \dfrac{1}{m_{11}}\left(m_{22}ur + \dfrac{1}{2}(m_{23} + m_{32})r^2 - d_{11}u + \tau_u + \tau_{\mathrm{eu}}\right) \\ \dot{v} = \dfrac{1}{m_{22}}(-m_{23}\dot{r} - m_{11}ur - d_{22}v - d_{23}r + \tau_v + \tau_{\mathrm{ev}}) \\ \dot{r} = \dfrac{1}{m_{33}}\left(-m_{32}\dot{v} - m_{22}vu - \dfrac{1}{2}(m_{23} + m_{32})ru + m_{11}uv - \right. \\ \qquad \left. d_{32}v - d_{33}v - d_{33}r + \tau_r + \tau_{\mathrm{er}}\right) \end{cases} . \tag{4}$$

2.2 Guidance

The guidance module aims to guide the USV along the desired path by calculating the current position of the USV and defining its relationship with the given path. As shown in Fig. 3, the USV virtual environment module can upload the desired path task $\boldsymbol{P}(\omega)$ as needed and the guidance sub-module calculates the desired course angle $\chi_d(t)$ and cross-track error $y_e(t)$ using the LOS guidance method. As for speed control, this paper focuses on path following, which is different from trajectory tracking and does not need to take temporal specification into account [27]. Therefore, the guidance module does not include speed interference. In our opinion, USV speed control could be directly achieved by the controller network with inputs such as speed error $u_e(t) = u_d(t) - u(t)$, heading error $\psi_e(t)$, and cross-track error $y_e(t)$, which will be detailed in the subsequent sections.

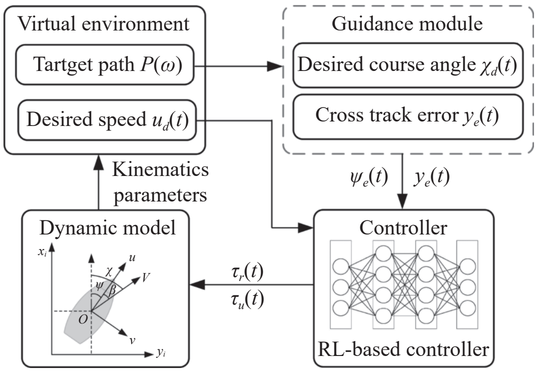


Fig. 3 Diagram of guidance module for a RL-based controller

LOS is a type of classical and effective navigation algorithm that reflects the helmsman's specific behavior in the accurate control of vessel navigation. A LOS vector is the line connecting a point mapped as seen from the current position of a vessel onto the tangent line of the desired path from that point to the current position of the vessel [28]. By keeping the vessel's resultant speed in alignment with LOS vector, the vessel is guided to approach the desired path. LOS guidance is independent of dynamics control, does not rely on any mathematical model, and needs few designed parameters. Moreover, it is insensitive to high-frequency white noise and uses the desired path and the vessel's real-time position only to determine the desired heading angle. In this way, the desired heading angle can be calculated efficiently in real-time.

This paper uses a LOS guidance algorithm based on the look-ahead distance to calculate the desired heading angle. The guidance principles are as shown in Fig. 4.

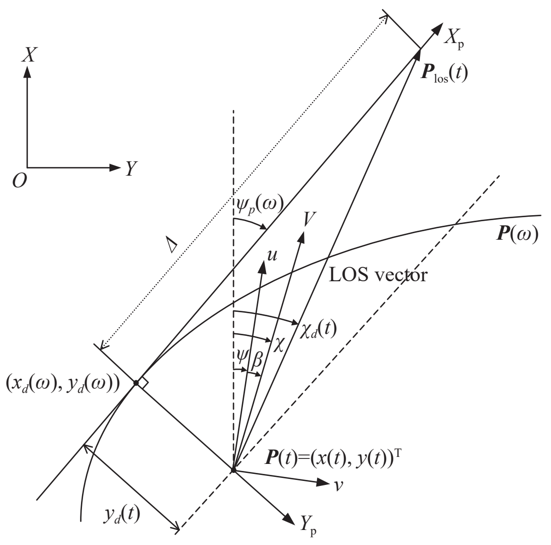


Fig. 4 LOS guidance based on the look-ahead distance

The LOS guidance algorithm is defined as

$$\chi_d(t) = \psi_p(\omega) + \arctan\left(-\frac{y_e(t)}{\Delta}\right) \tag{5}$$

where $\chi_d(t)$ is desired course angle; $\boldsymbol{P}(t)$ is the actual position of the USV; $(x_d(\omega), y_d(\omega))$ is desired position of the USV; $\psi_p(\omega)$ denotes the angle formed by the tangential line of the desired position on the desired course and X-axis; Δ is the look-ahead distance; $\boldsymbol{P}_{\rm los}(t)$ is a virtual guidance point; $y_e(t)$ is the cross-track error, such as the deviation of the current position of USV from the desired course. Therefore, the desired heading angle $\psi_d(t)$ is

$$\psi_d(t) = \chi_d(t) - \beta(t) = \psi_p(\omega) + \arctan\left(-\frac{y_e(t)}{\Delta}\right) - \beta(t). \tag{6}$$

The heading angle error is

$$\psi_e(t) = \psi_d(t) - \psi(t). \tag{7}$$

LOS guidance can be used for both straight and curved paths. Woo et al. [29] analyzed the switching principles of LOS guidance for a straight path. The switching principles were improved to achieve a smooth switch between straight and curved paths based on this guidance. If the USV mission path consists of several linked sub-paths of various lengths and shapes such as straight lines and arcs, the USV path following task follows the adjacent straight or curved sub-paths one by one. Moreover, it is assumed that this process is irreversible. In other words, the task does not return to the previous sub-path following even if USV turns back to the starting point.

It is assumed that from the starting point, the ith sub-path Path$_i$ starts at $\omega_{i-1}(x_{i-1}, y_{i-1})$ and ends at $\omega_i(x_i, y_i)$, $i \in N^*$. When Path$_i$ is straight, the USV will follow the straight path Path$_i$. At any point, the desired course angle could be displayed in a vector field, as shown in Fig. 5.

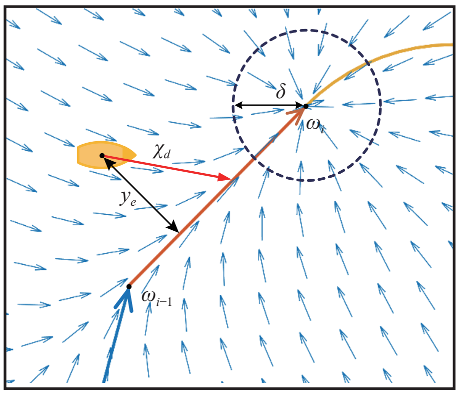


Fig. 5 Diagram of a straight LOS guidance vector field

Equation (5) can be used to calculate the desired course angle χ_d and the corresponding virtual guidance point $\boldsymbol{P}_{\rm los}$. When the USV enters a circle with the origin ω_i and a radius δ, it is believed that the task of following the current sub-path is completed, and it switches to the state of following the next sub-path. To ensure that the USV can be guided to the proximity of where the sub-path ends under extreme conditions, it is therefore defined that, if the virtual guidance point $\boldsymbol{P}_{\text{los}}$ is on an extension of a line segment $\omega_{i-1}\omega_i$, the virtual guidance point $\boldsymbol{P}_{\text{los}}$ of the USV will be fixed at the position where the sub-path ends ω_i.

When the USV follows a curved path, guidance principles are slightly different based on the sector in which the USV travels. As shown in Fig. 6, if the USV follows the curved path Path$_{i+1}$, the desired course angle at any point can be presented vividly in the vector field. Let O_c be the origin of the arcs that the sector formed by $\angle\omega_i O_c \omega_{i+1}$ is defined as Sector B, while the remaining area is divided equally into Sector A and Sector C with the origin O_c. Normally, the USV is in the normal state of path following when it is in Sector B. Considering that RL control could not be stable at the initial stage of learning, all possibilities must be taken into account to ensure the correct guidance for the USV at any position.

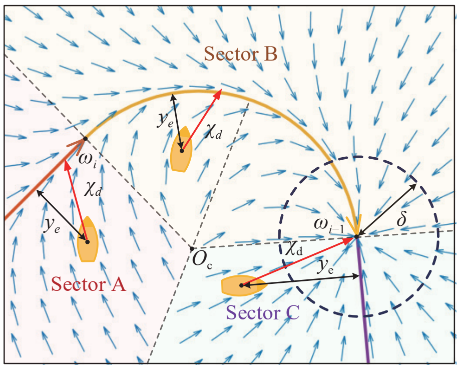


Fig. 6 Diagram of a curved LOS guidance vector field

The policy for different conditions is as follows: when the USV is in Sector A, LOS guidance is implemented to guide the USV along a sub-path path$_i$; when the USV is in Sector B and the virtual guidance point $\boldsymbol{P}_{\text{los}}$ is in Sector B, LOS guidance is conducted to guide the USV along the current sub-path path$_{i+1}$; when USV is in Sector B and virtual guidance point $\boldsymbol{P}_{\text{los}}$ enters Sector C, a virtual guidance point will be fixed at the position where the sub-path ends ω_{i+1}; when USV is in Sector C, the virtual guidance point $\boldsymbol{P}_{\text{los}}$ will be also fixed at ω_{i+1}.

The guidance module will output the heading angle error $\psi_e(t)$ and cross-track error $y_e(t)$. Moreover, $\psi_e(t)$ can be calculated with (7), while the value of $y_e(t)$ could be calculated considering the geometrical relationship at any time.

When the USV follows a straight path or is in Sector A or Sector C of a curved path, $y_e(t)$ is the Euclidean distance from the USV to the straight line.

$$y_e(t) = \sqrt{(x(t) - x_d(\omega))^2 + (y(t) - y_d(\omega))^2} \tag{8}$$

When the USV follows a curved path and is in Sector B, let the radius of the arc be r, and the origin $O_c(x_o, y_o)$, so that $y_e(t)$ is the shortest Euclidean distance from USV to the arc:

$$y_e(t) = r - \sqrt{(x(t) - x_o)^2 + (y(t) - y_o)^2}. \tag{9}$$

2.3 Reward function

The reward function defines the target in RL [30]. In each time step, the environment assesses the performance of the USV in the execution of the path following task, and give a reward output. The objective of the RL controller is to maximize the total reward for the long term. Hence, the reward function requires clearly defines the quality of the USV state, and plays a significant role in RL. On this basis, several independent performance indices have been designed, including heading reward, distance reward, and speed reward, to measure the effect of path following, respectively. Moreover, these indices are weighted according to actual needs to achieve comprehensive assessment.

Index 1 Heading reward. To make the USV move forward in the correct direction, a maximum heading reward is obtained when the absolute value of the USV heading angle error $|\psi_e|$ is 0, but the heading reward should decrease rapidly when $|\psi_e|$ deviates from 0. The heading reward is defined as

$$r_{\psi_e} = 2\mathrm{e}^{-k_\psi|\psi_e|} - 1 \tag{10}$$

where $r_{\psi_e} \in (-1, 1]$, and k_ψ is an adjustment coefficient used to determine the steepness of the reward at the peak.

Index 2 Distance reward. To make the USV move along a given path, a distance reward r_{y_e} is proposed. A distance reward is defined as

$$r_{y_e} = 2\mathrm{e}^{-k_y|y_e|} - 1 \tag{11}$$

where $r_{y_e} \in (-1, 1]$, and k_y plays a role in similarly adjusting the reward curve.

Index 3 Speed reward. To ensure the movement of the USV at the desired speed, a speed reward r_{u_e} is proposed as

$$\begin{cases} r_{u_e} = p r_{u1} + (1-p) r_{u2} \\ r_{u1} = 2\mathrm{e}^{-k_{u1}|u_e|} - 1 \\ r_{u2} = 2\mathrm{e}^{-k_{u2}|u\cos(\psi_e) - u_d|} - 1 \end{cases} \tag{12}$$

where $r_{u_e} \in (-1, 1]$ and r_{u_e} consists of r_{u1} and r_{u2}, respectively. p is a weight coefficient and $p \in [0, 1]$. k_{u1} and k_{u2} are the adjustment coefficients of r_{u1} and r_{u2}, respectively. Among them, r_{u1} is used to constrain the speed of the USV and $u\cos\psi_e$ in r_{u2} indicates the projection of the USV speed onto the desired course; the speed can reach the maximum only if the projection speed on the desired course is identical to the desired speed.

The original function of speed reward r_{u_e} is consistent with (10) and (11) such as $r_{u_e} = 2\mathrm{e}^{-k_u|u_e|} - 1$, but some shortcomings were noticed in the definition during the training. The above equation can only achieve a constraint on USV speed. Compared with the required accuracy of heading and course following, speed control is easier to achieve. Therefore, the controller often gives priority to speed reward at the initial stage of training, but overlooks heading and distance rewards. Meanwhile, the USV controller normally has an unstable output of rudder angle during the initial stage of training. In addition to the influence of USV steering inertia, it is more difficult to achieve r_{ψ_e} and r_{y_e}, which allows the controller network to become trapped by a local optimal reward easily. To resolve this problem, the speed reward r_{u_e} is connected with the heading reward to ensure a similar difficulty in obtaining r_{ψ_e}, r_{y_e}, and r_{u_e}.

After such improvement, speed reward r_{u_e} can reach the peak only if the USV moves forward along the desired heading at the desired speed.

Composite index Reward function. In summary, the final reward expression is given as

$$r = w_{\psi} r_{\psi_e} + w_y r_{y_e} + w_u r_{u_e} \tag{13}$$

where w_{ψ}, w_y, and w_u are the weights of r_{ψ_e}, r_{y_e}, and r_{u_e}, respectively. These indices could be adjusted dynamically according to the actual needs of a mission to evaluate the reward of the action comprehensively.

3. USV path dataset

As an important part of RL, environment modeling is the simulation of the environment response mode [30]. Nevertheless, researchers often do not conduct extensive environment modeling because complex and varying tasks exist in path planning. Studies related to RL control with marine vessels normally focus on the simple paths [22,29]. Moreover, these test scenarios are generally mentioned in parts of simulations in the papers. Researchers placed an agent in a fixed scenario and then trained an RL controller from scratch (the RL controller was trained separately for each test scenario). The control errors at different stages of training were compared to verify the effectiveness of the RL algorithm [31]. But an RL controlled trained in a single scenario may easily cause over-fitting and lead to network degeneration. For engineering applications, such limited navigation experience for DRL controller is far from enough. When facing extreme scenarios, the controller network's output is likely to be unpredictable without enough training experience. It will cause uncertainty and hidden safety problems in the engineering applications

There has to be a balance between the performance and the generalization of the environment. When training to a few particular environment variations, the deep policy tends to overfit and loses the ability to generalize to different paths. Therefore, it must be mentioned that the random generation of paths, allows a distributed stochastic sampling of the problem's domain that enhances the generalization of the task at the cost of a larger training process. A path dataset was constructed to satisfy the needs of multiple tasks.

When considering the needs of a USV RL control, it is necessary to construct a variety of USV mission paths, which must cover as many path scenarios as possible and serve as a baseline for comparing the performance of different algorithms. Therefore, two types of path datasets were constructed, the random path dataset and the standard path dataset.

First, the random path dataset plays the role of providing all kinds of task scenarios for RL controller training, which meet the following requirements:

(i) It can be generated quickly with an automation script.

(ii) Paths should contain straight lines and curves simultaneously, and the boundary between paths should be continuous.

(iii) Paths must not go beyond the given boundary and be easy to display visually.

Second, the standard path dataset must meet the following requirements:

(i) The path should be typical and cover a variety of conditions.

(ii) The dataset should contain a single straight line, curved and mixed paths, and facilitate the quantitative and comparative control analysis of diverse paths.

A flowchart of the automated path generation algorithm designed in this paper is shown in Fig. 7 and Fig. 8, and Fig. 9 shows the instance diagram of the random path dataset.

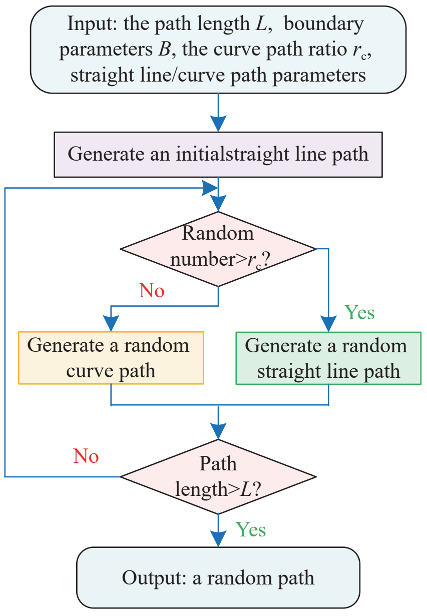
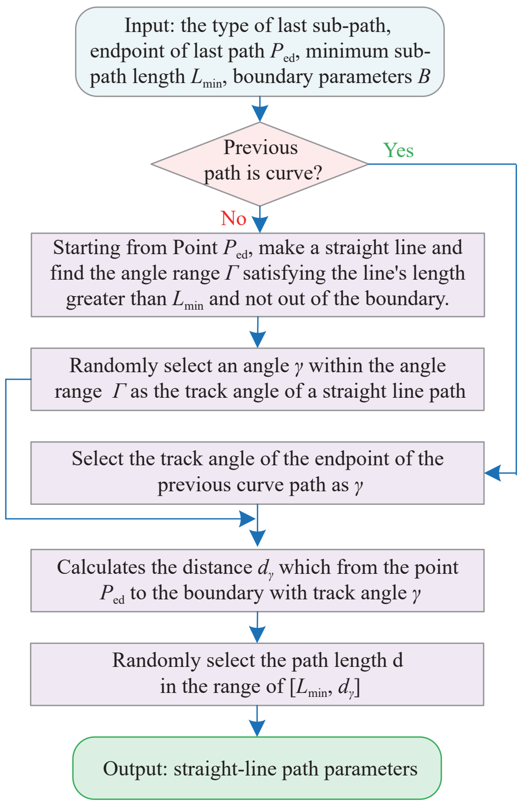

Input: the path length L, boundary parameters B, the curve path ratio r_c, straight line/curve path parameters
Generate an initialstraight line path
Random number>r_c?
No
Yes
Generate a random curve path
Generate a random straight line path
No
Path length>L?
Yes
Output: a random path

Fig. 7 Flowchart for generating a random path


Input: the type of last sub-path, endpoint of last path P_{ed}, minimum sub-path length L_{min}, boundary parameters B
Previous path is curve?
Yes
No
Starting from Point P_{ed}, make a straight line and find the angle range Γ satisfying the line's length greater than L_{min} and not out of the boundary.
Randomly select an angle γ within the angle range Γ as the track angle of a straight line path
Select the track angle of the endpoint of the previous curve path as γ
Calculates the distance d_γ which from the point P_{ed} to the boundary with track angle γ
Randomly select the path length d in the range of $[L_{min}, d_\gamma]$
Output: straight-line path parameters
(a) Detailed implementation of random straight-line path

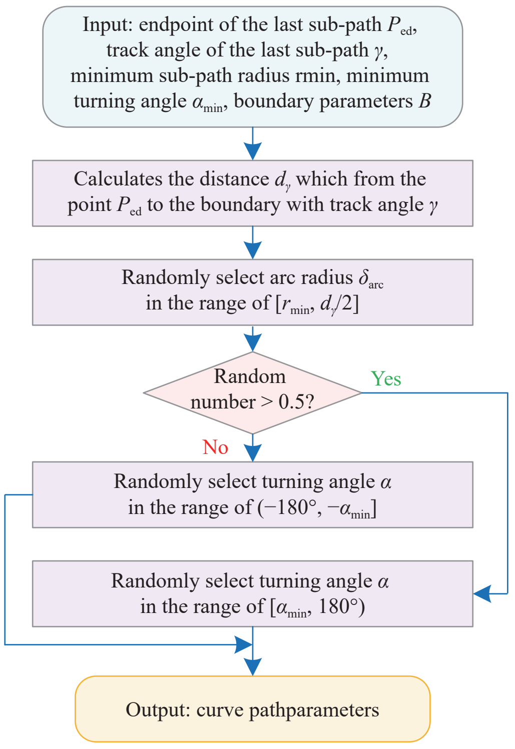

Input: endpoint of the last sub-path P_{ed}, track angle of the last sub-path γ, minimum sub-path radius rmin, minimum turning angle α_{min}, boundary parameters B
Calculates the distance d_γ which from the point P_{ed} to the boundary with track angle γ
Randomly select arc radius δ_{arc} in the range of $[r_{min}, d_\gamma/2]$
Random number > 0.5?
Yes
No
Randomly select turning angle α in the range of $(-180°, -\alpha_{min}]$
Randomly select turning angle α in the range of $[\alpha_{min}, 180°)$
Output: curve pathparameters
(b) Detailed implementation of random curve path

Fig. 8 **Detailed implementation of the random straight-line path and random curve path**

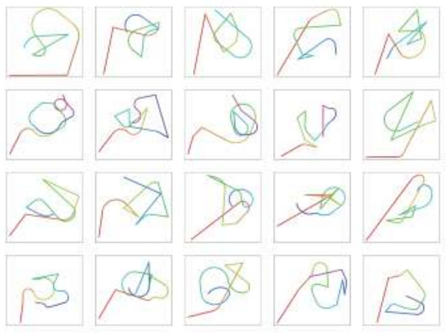


Fig. 9 **Instance diagram of random path dataset**

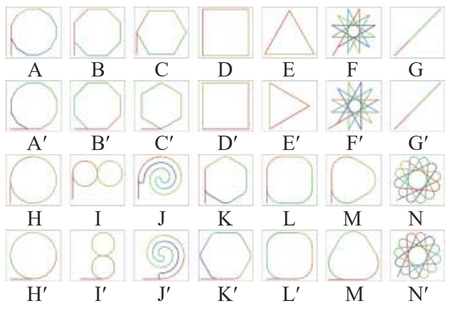


Fig. 10 **Dataset of 28 standard paths**

To meet the above requirements, 14 different mission paths are designed (Fig. 10), including seven polygonal paths (paths A−G), three curved paths (paths H−J) and four mixed paths (paths K−N). Each polygonal path was comprised of sub-paths of equal length, which are connected with a fixed steering angle. Among them, Path G is a back-and-forth path on a straight line. A curved path is formed by an arc or spiral, while a mixed path is composed of straight lines and arcs connected alternately. Based on this, paths were mirrored to form 28 types of horizontally symmetrical routes, eventually measuring an RL controller's performance more comprehensively.

To thoroughly test each type of path, all standard paths were set to depart from the lower-left corner along a start line in red and then repeat the closed path many times until the accumulated distance reached the preset mileage of each path. The average performance index of each control algorithm with the same reference path was compared to objectively and comprehensively evaluate these algorithms.

Fig. 11 details the composition and interaction of modules under the step function in the Gym environment customized in this paper.

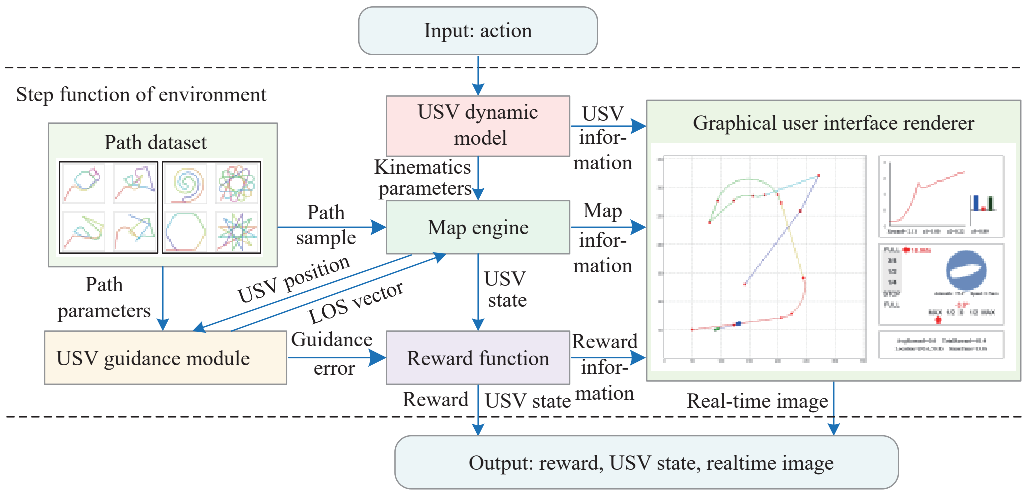


Fig. 11 Composition and interaction of modules under a step function in a customized Gym environment

4. DRL based controller

As an important part of machine learning, reinforcement learning means an agent learns the action policy needed to fulfill the mission through continuous interaction with the environment. While performing a mission, the agent interacts with the environment to generate a new state, while the environment grants a reward. An RL algorithm uses an action policy to interact with the environment, generate a reward, and then apply the reward in modifying the action policy. After several iterations, the agent will learn the action policy needed to complete the mission.

The traditional RL approach is often limited to a finite and discrete action space and sample space. This makes it difficult to process more complex missions close to the actual conditions, often involving the continuous and high-dimensional state spaces and action spaces. DRL combines RL with deep learning capability to process high-dimension data, while deep Q networks (DQN) must be the first successful product of such a combination [31]. Scholars have extensively studied the RL approach in recent years and proposed more and newer DRL methods such as asynchronous advantage actor-critic (A3C) [32], deep deterministic policy gradient (DDPG) [33], and PPO [34].

USV path following requires considering paths of complex shapes; the entire process of guidance is highly sequential and involves interconnected trajectories. Therefore, this paper combines the PPO framework [35] with the LSTM with a strong ability to process temporal data and then proposes a LSTM-DPPO based path planning algorithm, which allowed several environments to gather the experience at the same time by parallel central processing unit (CPU) process, and the training work is divided among the GPUs.

4.1 PPO Algorithm

PPO is a model-free, on-policy, actor-critic, policy-based method. As an improved algorithm, it originated from a trust region policy optimization (TRPO) algorithm [36]. The standard solution of a TRPO algorithm requires an enormous amount of calculation, but PPO is a first-order approach and simplifies the optimization process, which reduces the calculation effort considerably and achieves a good effect.

In the process of training, the state of the agent at the time t is denoted by s_t. PPO uses stochastic policy, so that the policy π could be modeled as $\pi(a|s)$; that is, the probability distribution of a at the state s. In this paper, π is the LSTM controller network, whose parameter is denoted by θ. The controller network π_θ input the state of the agent s_t and outputs a gaussian probability distribution of actions. Sampling is further conducted to obtain a_t. Meanwhile, the environment will grant a reward based on s_t and a_t, that is, $r_{t+1}(s_t,a_t)$. The objective of RL is to find the optimal policy π^* to maximize the accumulated discount reward R_t as

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{k+1} \tag{14}$$

where γ is a discount factor and $\gamma \in [0,1]$.

The policy π is evaluated by two pecspective, the state-value function $V^\pi(s)$ and action-value function $Q^\pi(s,a)$. $V^\pi(s)$ is the expected return when starting in s and following π thereafter. Meanwhile, $Q^\pi(s,a)$ is the expected return starting from s, taking the action a, and thereafter following policy π:

$$V^{\pi}(s_t) = \mathrm{E}_{\pi}[R_t|s_t] = \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{k+1}|s_t\right], \tag{15}$$

$$Q^{\pi}(s_t,a_t) = \mathrm{E}_{\pi}[R_t|s_t,a_t] = \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{k+1}|s_t,a_t\right]. \tag{16}$$

An advantage function $A^{\pi}(s,a)$can be used to determine the advantage achieved by choosing an action a at the state s when compared with other actions. If the action a is a better choice.

$$A^{\pi}(s_t,a_t) = Q^{\pi}(s_t,a_t) - V^{\pi}(s_t) \tag{17}$$

The PPO algorithm offers a truncated version of generalized advantage estimation [37] to estimate an advantage function, which could significantly reduce the variance while maintaining a tolerable level of bias. Therefore, $\hat{A}_t$ is defined as the estimator of the advantage function at the timestep t, and T represents the number of time steps taken continuously under the policy π in an episode (T is much shorter than the episode). The expression of $\hat{A}_t$ is as follows:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T). \tag{18}$$

A policy gradient algorithm is an RL algorithm for policy-based search. It estimates a policy gradient and applies it in a stochastic gradient ascent. Meanwhile, $L^{\mathrm{PG}}(\theta)$ represents the policy gradient loss, and $\hat{\mathrm{E}}_t$ serves as empirically obtained estimates of the expectation. The expression is as follows:

$$L^{\mathrm{PG}}(\theta) = \hat{\mathrm{E}}_t[\log \pi_{\theta}(a_t|s_t)\hat{A}_t]. \tag{19}$$

However, empirically, the gradient obtained by using the expression above is likely to lead to too large policy updates. By optimizing the surrogate objective function $L(\theta)$, TRPO could constrain the amplitude of the policy update.

$$L(\theta) = \hat{\mathrm{E}}_t\left[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}\hat{A}_t\right] \tag{20}$$

On this basis, PPO includes two simpler realizations, i.e., clipped surrogate objective and adaptive KL penalty coefficient. The former has a better effect, so that method is employed in this paper. Its objective function is $L^{\mathrm{CLIP}}(\theta)$.

$$\mathrm{Ratio}_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}, \tag{21}$$

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathrm{E}}_t[\min(\mathrm{Ratio}_t(\theta)\hat{A}_t, \mathrm{clip}(\mathrm{Ratio}_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)], \tag{22}$$

where $\mathrm{Ratio}_t(\theta)$ is probability ratio, and ε is a hyperparameter. The clip function is used to restrict the $\mathrm{Ratio}_t(\theta)$ within the interval of $[1-\varepsilon, 1+\varepsilon]$. Therefore, the degree of policy gradient update is effectively constrained from the TRPO first-order approach.

4.2 LSTM-DPPO network framework

DPPO is a distributed version of PPO [38]. Multiple workers have used to collect experience in USV virtual environments independently. The mission path in each virtual environment is sampled from a random path dataset so that DPPO could effectively prevent the correlation of experience, achieving a significantly better learning effect for DPPO than for PPO.

Meanwhile, LSTM-DPPO is based on the actor-critic algorithm, including policy and value networks with similar network structures. The policy network has state space sequence $\boldsymbol{S}$ as its input and a probability distribution parameter of action as its output. In the process of training, a policy gradient is employed to learn the control policy π_{θ} with parameters θ. The value network has the same input as the policy network, but outputs $V_{\phi}(\boldsymbol{S})$, the estimated value of the state space sequence $\boldsymbol{S}$. In practice, sample rollout is often conducted to approximate the parameter ϕ of the value network $V^{\pi_{\theta}}(\boldsymbol{S})$. The specific definition is given as follows: State and action spaces at the time t are defined as $\boldsymbol{S}_t = \left\{\psi_e^t, \dot{\psi}_e^t, y_e^t, \dot{y}_e^t, u_e^t\right\}$ and $\boldsymbol{A}_t = \left\{a_p^t, a_s^t\right\}$, respectively. The meanings and ranges are presented in Table 1.

Table 1 Constraints and ranges for variables in state and action spaces

Variable	Meaning	Range
ψ_e	Heading angle error	[−180,180]
$\dot{\psi}_e$	Heading angle error derivative	$(-\infty,+\infty)$
y_e	Cross-track error	$(-\infty,+\infty)$
$\dot{y}_e$	Cross-track error derivative	$(-\infty,+\infty)$
u_e	Speed error	[−20,20]
a_p	Power output	[−1,1]
a_s	Steering output	[−1,1]

The sequence of state space within the range in T timesteps starting from the time t is defined as $\boldsymbol{S} = \{\boldsymbol{S}_t, \boldsymbol{S}_{t-1}, \boldsymbol{S}_{t-2}, \cdots, \boldsymbol{S}_{t-T+1}\}$. The diagram of policy network structure is presented in Fig. 12.

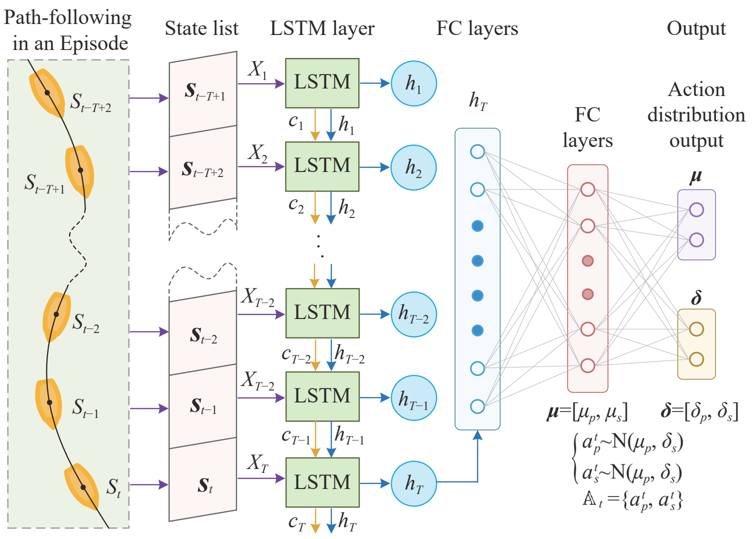

Path-following in an Episode
State list
LSTM layer
FC layers
Output
Action distribution output
FC layers
h_T
LSTM
μ
δ
$\mu=[\mu_p, \mu_s]$
$\delta=[\delta_p, \delta_s]$
$a_p^t \sim N(\mu_p, \delta_s)$
$a_s^t \sim N(\mu_p, \delta_s)$
$\mathbb{A}_t=\{a_p^t, a_s^t\}$

Fig. 12 Diagram of the policy network structure.

Fig. 12 demonstrates how a policy network outputs a probability distribution parameter. It is defined that τ is a trajectory formed by a USV performing a path following mission in the environment $(s_1, a_1, s_2, a_2, \cdots, s_{t-1}, a_{t-1}, s_t, a_t)$. At the time t, T past state space sequences $\boldsymbol{S}$ are intercepted forward. The elements in S are input into the LSTM network in chronological order. After repeating for T times, the output of recurrent neural networks h_T is obtained. h_T is a group of one-dimension arrays with the length l, and l is the number of LSTM elements in the network. After passing the fully connected (FC) layer, h_T outputs the distribution of action. It is assumed that the action is subject to a normal distribution, and the network output distribution has its average defined as $\boldsymbol{\mu} = [\mu_p, \mu_s]$ and variance as $\boldsymbol{\delta} = [\delta_p, \delta_s]$. In the process of training, the action value of an agent is selected through action distribution sampling, so as to ensure that the action of an agent is selected in a random and explorative way. In the progress of training, the network converges with the variance $\boldsymbol{\delta}$ decrease gradually. After network training is completed, the variance $\boldsymbol{\delta}$ will be ignored in the practical configuration of the controller, but average $\boldsymbol{\mu}$ is used as the action value.

The LSTM layer and a fully connected layer of a value network have the same structure as a policy network, but the output layer outputs only a single variable, i.e., a state sequence value $V_\phi(\boldsymbol{S})$.

The proposed LSTM-DPPO algorithm consists of workers and a master. The workers have several instances, which run in separate processes. The workers are responsible for collecting data from experience, calculating the gradients of the policy and value networks, updating the network weights locally, transmitting local weights to the master, and receiving the network weights from the master to replace the old local weights. This process is repeated continuously until the end of the training. The master assumes the responsibility of scheduling and managing workers, summing up the weights transmitted from workers, calculating their average weights as the global weights, and sending the latest global weights to the workers. Algorithm 1 and Algorithm 2 describe the realization process of LSTM-DPPO.

Algorithm 1 LSTM-DPPO (Master)

```
Input:
Number of epochs $N$; the number of workers $W$
Output:
Policy network parameter $\theta$, value network parameter $\phi$
1: Initialize the global policy network $\pi_\theta$ and value network $V_\phi$ with empty weights.
2: for $i \in \{1,2,\cdots,N\}$ do
3:     for $j \in \{1,2,\cdots,W\}$ do
4:         Collect the weights $\theta_j$ and $\phi_j$ by worker $j$
5:     end for
6:     Calculate the averaged weights $\bar{\theta}$ and $\bar{\phi}$ from workers' weights
7:     Assign averaged weights to global policy network $\pi_\theta$ and value network $V_\phi$
8:     for $j \in \{1,2,\cdots,W\}$ do
9:         Assign weights of networks to worker $j$
10:    end for
11: end for
```

Algorithm 2 LSTM-DPPO (Worker)

```
Input:
Number of epochs $N$; rollout size $R$; time step size $T$; discount factor $\gamma$; the number of sub-iterations with policy
```

```
and value updates $N_P, N_V$, respectively; learning rate $l_r$.
**Output**:
Weights of local policy network $\theta$ and value network $\phi$.
1: Initialize the USV environment, load the path data set, randomly initialize the local policy network $\pi_\theta$ and value network $V_\phi$ with weights $\theta$ and $\phi$.
2: for $i \in \{1,2,\cdots,N\}$ do
3:     Receive network weights $\pi_{\theta\text{new}}$, $V_{\phi\text{new}}$ from the master, assign weights to the local network: $\pi_\theta \leftarrow \pi_{\theta\text{new}}$, $V_\phi \leftarrow V_{\phi\text{new}}$
4:     for $j \in \{1,2,\cdots,R/T\}$ do
5:         Run policy $\pi_\theta$ for $T$ time steps in the USV Gym environment, collecting $\{s_t,a_t,r_t\}$ for $t \in \{(j-1)T,\cdots,jT-1\}$
6:         Calculate return $\hat{R}_t = \sum_{t=(j-1)T}^{jT-1} \gamma^{t-(j-1)T} r_t + \gamma^{\mathrm{T}} V_\phi(s_{jT})$
7:         Calculate estimated advantages $\hat{A}_t = \hat{R}_t - V_\phi(s_t)$
8:         Store partial trajectory information
9:     end for
10:    $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
11:    for $p \in \{1,2,\cdots,N_P\}$ do
12:        $\text{Ratio}_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$
13:        $L^{\text{CLIP}}(\theta) = \hat{\mathrm{E}}_t\left[\min(\text{Ratio}_t(\theta)\hat{A}_t, \text{clip}(\text{Ratio}_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right]$
14:        Calculate gradient $\nabla_\theta L^{\text{CLIP}}$ and update $\theta$ with the learning rate $l_r$
15:    end for
16:    for $v \in \{1,2,\cdots,N_V\}$ do
17:        $L^{\text{BL}}(\phi) = -\sum_{t=1}^{T} (\hat{R}_t - V_\phi(s_t))^2$
18:        Calculate gradient $\nabla_\phi L^{\text{BL}}$ and update $\phi$ with learning rate $l_r$
19:    end for
20:    Send local weights $\theta$ and $\phi$ to master
21: end for
```

5. Simulations

To verify the proposed USV path following algorithm, the random path dataset was employed to train the LSTM-DPPO controller network. First, the LSTM-DPPO controller at different training stages is extracted to analyze the algorithm's self-learning capability; then, the path following performance of the controller is tested in the standard path dataset. Subsequently, the angle and distance errors, as well as the average rewards, are calculated to compare the LSTM-DPPO algorithm with other control algorithms in terms of tracking performance. Ultimately, to verify the contribution of the path dataset to the control performance, the LSTM-DPPO algorithm is studied with different volumes of the dataset. Various scales of path samples are intercepted from the random path dataset to train the controller network from scratch, which is tested by the standard path dataset.

The USV controller is trained in a server with two E5-2678 V3 CPUs and four RTX 2080Ti GPUs. The parameters for the LSTM-PPO algorithm are configured as follows. The number of workers is eight (two working processes for each graphic card). The number of training epochs was set to 1 000; the rollout size is selected as 8 192. The LSTM input time step range T is selected as 30. A discount factor was selected as 0.99. The number of sub-iterations with policy and value updates is set to the default value of ten. For the network training, an Adam optimizer is used to train both the policy and value networks. The learning rate is set to 10^{-4}. The USV environment parameters are configured as follows: simulation timestep is defined as 0.1 s (10 Hz). The path length L is set to 2 km. The boundary length is selected as 400 m. The curved path ratio r_c is selected as 0.4. The sample size of the random path dataset is set to 1000. The desired speed in each training epoch was generated randomly between 2−8 m/s. Reward adjustment coefficients k_ψ, k_y, k_{u1}, and k_{u2} are 0.1, 0.1, 0.2, and 0.2, respectively. Reward weights w_ψ, w_y, w_u, and p are 1/3, 1/3, 1/3, and 0.7, respectively.

$$\begin{cases} \boldsymbol{M} = \begin{bmatrix} 1\,200 & 0 & 0 \\ 0 & 1\,500 & 0 \\ 0 & 0 & 450 \end{bmatrix} \\ \boldsymbol{D} = \begin{bmatrix} 400 & 0 & 0 \\ 0 & 600 & 0 \\ 0 & 0 & 300 \end{bmatrix} \end{cases}$$

Fig. 13 presents the training curves of the LSTM-DPPO controller with a random path dataset. The accumulated reward in the first 230 epochs increased monotonically in an S-shaped curve after smoothing, and soared from −1 420 to 810. Subsequently, it oscillated around 800. The accumulated reward does not become stable at the late stage of training, but constantly oscillates within a favorable interval. Through the USV visualized environment, it can be seen that the desired speed stochastically causes the oscillation of the accumulated reward. When the desired speed is lower, it is easier to control the USV, and the path following error is smaller. In contrast, if the desired speed is larger, the USV needs to frequently adjust the state for path following under the limited maneuverability, causing a more significant path following error, so that the accumulated reward is poorer.

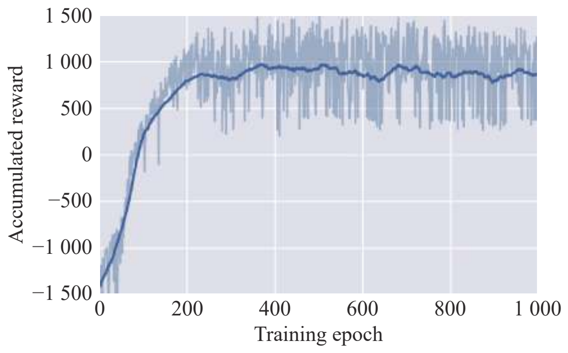


Fig. 13 Training curve of the LSTM-DPPO

To analyze the self-learning capability of the LSTM-DPPO algorithm, the weights of the policy network at different stages of training are extracted to test for all the standard path dataset. Square and rounded rectangular paths are taken as the typical polygonal and curved paths for analysis and illustration. The test parameters are set as follows: USV initial position is (−30,−30); desired speed is 5 m/s; the initial heading angle is given randomly; the maximum operation epoch is 3 000 steps.

Fig. 14 and Fig. 15 present the path following trajectory of the LSTM-DPPO algorithm on typical polygonal and curved paths at different stages of training. Fig. 14(a) and Fig. 15(a) show the result at the beginning of training, that is, the USV turned in a circle at the origin. As the training epochs increases, the USV learns to follow the mission path stably, but can not control its speed accurately. The speed is so low that the USV can not reach the endpoint of the path within the specified time steps in the episode (see Fig. 14(b) and Fig. 15(b). After 200 training epochs (see Fig. 14(c) and Fig. 15(c)), the USV can travel to a farther position along the mission path, but large errors could be still observed at the turning position of the polygonal path (see Fig. 14(c)). Compared with the polygonal path, the curved path has a much smaller following error (see Fig. 15(c)). It can be explained that the curved path is smooth and continuous, causing the continuous variation of the USV heading error angle, but the discontinuity of the polygonal path caused an abrupt change of the USV heading error angle, resulting in a large tracking error for the USV at the turning positions of the polygonal path. After 1 000 training epochs, the USV can finish all the paths within the specified time steps, and the tracked trajectory reached the ideal state (see Fig. 14(d) and Fig. 15(d)). After observing Fig. 14(d), it is found that the USV had a more continuous and accurate trajectory at the turning position. Fig. 16 and Fig. 17 present the heading angle error and cross-track error curves corresponding to the typical polygonal and curved paths, respectively. The convergence rates of both heading angle errors and cross-track errors have been significantly improved along with the increase of training epochs.

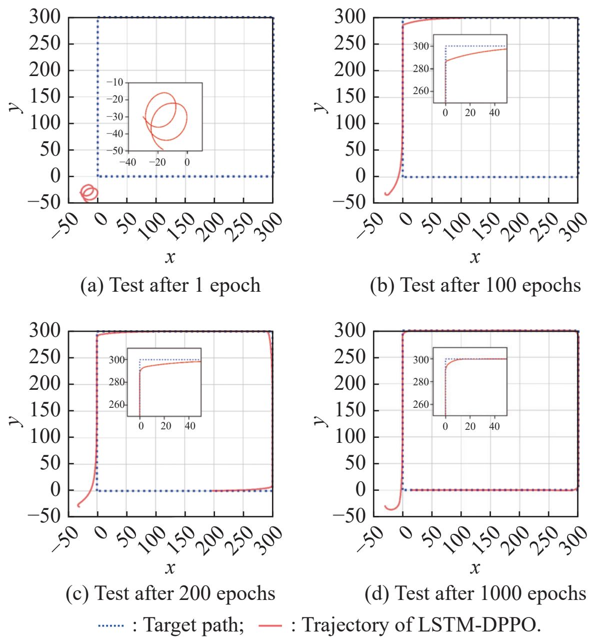


Fig. 14 Trajectory of a typical polygonal path following with LSTM-DPPO

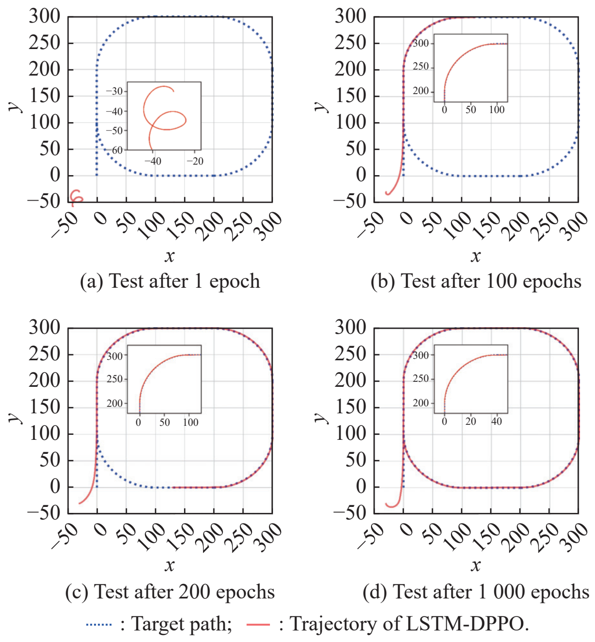


Fig. 15 Trajectory of a typical curve path following with LSTM-DPPO

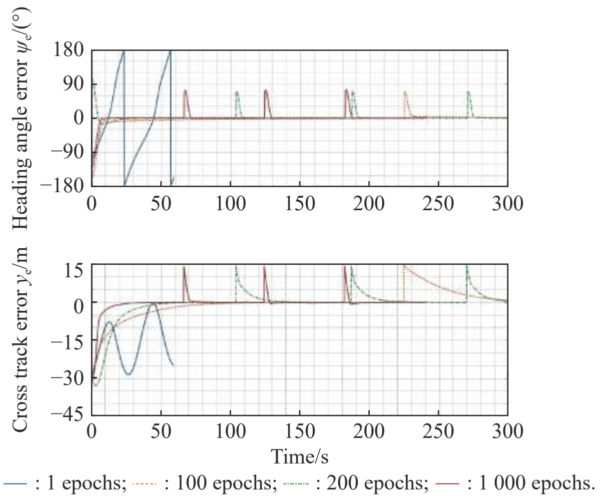


Fig. 16 History of the heading angle and cross-track errors during the typical polygonal path following experiments with various training epochs

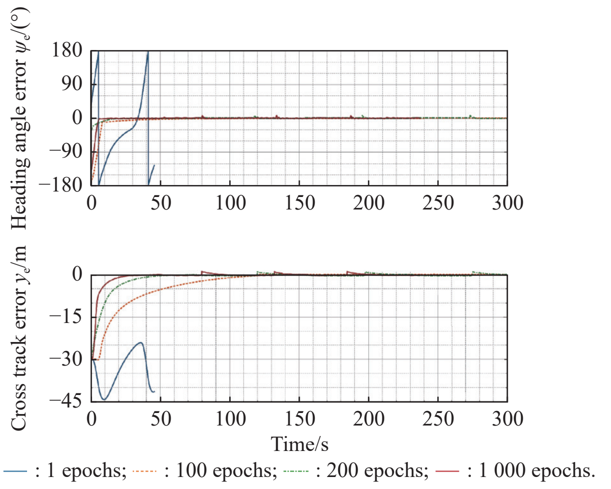


Fig. 17 History of the heading angle and cross-track errors during the typical curved path following experiments with various training epochs

To further analyze the advantages and disadvantages of the LSTM-DPPO algorithm when compared with other algorithms, three control methods were selected for comparison, including an MLP-PPO algorithm (with LSTM layer deleted but other parameters unchanged compared with LSTM-DPPO algorithm), a classic PID control algorithm, and a human player using a joystick for USV control. Fig. 18 and Fig. 19 present four different trajectories of following the typical polygonal and curved paths. The trajectories of path following with four methods including LSTM-DPPO, MLP-PPO, PID and human player, are indicated in red, purple, yellow, and green, respectively. As revealed in Fig. 18, these four control methods can ensure the stable following of a straight path, but their differences exist mainly in the control of turning performance. The enlarged detail in the upper right corner of Fig. 18 clearly shows that a well-trained LSTM-DPPO controller has the best performance since it could achieve rapid convergence of trajectory error. The MLP-PPO controller had a slower response and larger trajectory error. The PID controller faces overshoot and oscillatory convergence in the trajectory. The joystick is used to manipulate the USV throttle and rudder angle for manual path following control along each path in the standard path dataset. As shown in the green trajectory in Fig. 18, the human player can achieve a good effect by controlling the USV under the brain-image-joystick feedback mechanism. Fig. 19 shows the effect of following various control methods on a curved path. After observing the trajectories of following a curved path, it is found that LSTM-DPPO and MLP-PPO could control the cross-track error to a small range; the PID algorithm lag behind to some extent, and its trajectory is normally at the external edge of the turning position. Control of following the curved path was fairly challenging when using a joystick, which had the most significant path following error.

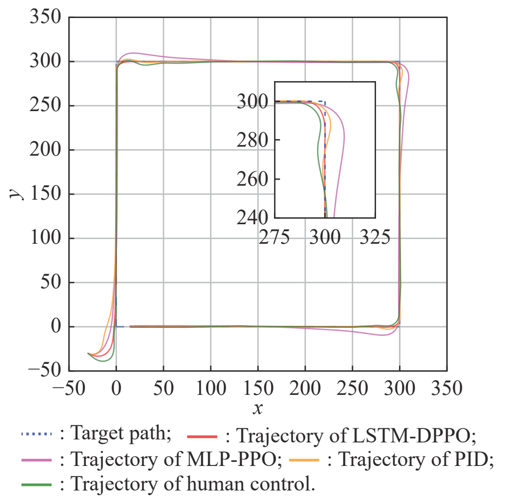


Fig. 18 Trajectories of following the typical polygonal path with four control methods

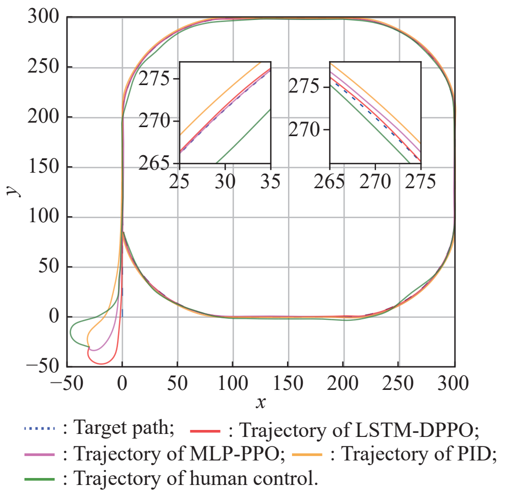


Fig. 19 Trajectories of following the typical curved path with four control methods

Fig. 20 and Fig. 21 contain the heading angle and cross-track error curves of the four control methods with regard to the two typical paths. The LSTM-DPPO algorithm could control the angle and distance errors faster than the other methods, and reach the ending point within the shortest period.

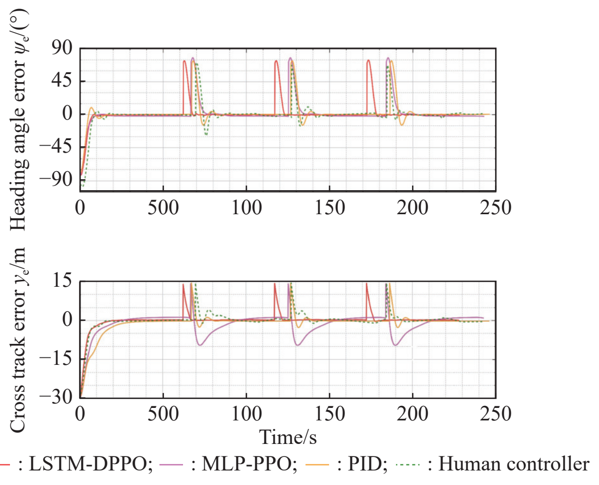


Fig. 20 History of the heading angle and cross-track errors during the typical polygonal path following experiments with four control methods

Table 2 shows the performance indices of various control methods in the path following test with a standard path dataset. There are three test conditions: a polygonal path, a curved/mixed path, and a combination of all paths above. The test indices include the root mean square (RMS) of heading angle error (H.A.E.), cross-track error (C.T.E.), and corresponding average reward (accumulated reward/simulation step size). The bolded item in each row represents the optimal value which is the minimum error. As revealed in Table 2, the average reward of the LSTM-DPPO method goes up constantly with the increase in training step size. Compared with the three other control methods, LSTM-DPPO performed better in all the performance parameters under a polygonal path test condition. In the curved/mixed path, the average reward was basically consistent among the LSTM-DPPO, MLP-PPO and PID methods. It can be interpreted that the training dataset is a combination of curves and straight lines, including sudden turns and even reversals. These features are similar to the polygonal path in the test dataset, so a better simulation result can be obtained in the polygonal path. However, the curved/mixed path in the test dataset is dominated by smooth and continuous curves, which are different from the training dataset in terms of path features, but the DRL controller still performs at a good level even though its performance is slightly inferior to PID controller. From the perspective of all the standard path dataset, the proposed algorithm has the remarkable capability to adapt to different paths and achieve the superior control performance of path following.

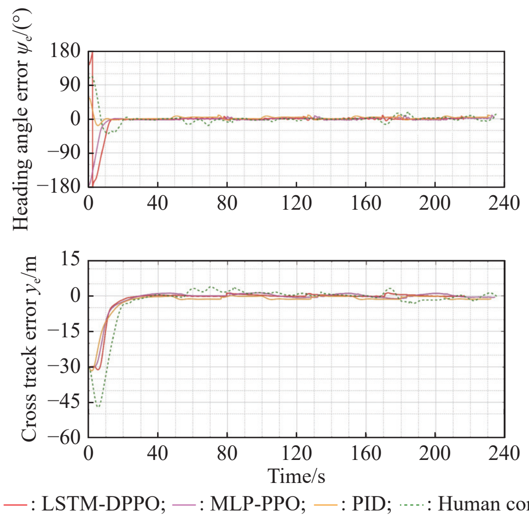


Fig. 21 History of the heading angle and cross-track errors during the typical curved path following experiments with four control methods

Table 2 Comparison results of the RMS for H.A.E. and C.T.E. during the path following validation

Performance indices using standard path datasets		LSTM-DPPO100 epochs	LSTM-DPPO 200 epochs	LSTM-DPPO 1 000 epochs	MLP-PPO	PID	Joystick
Polygonal path	RMS H.A.E/(°)	20.27	20.63	**11.19**	16.57	19.91	21.11
	RMS C.T.E/m	3.78	3.38	**3.12**	3.80	4.49	5.26
	Average reward	0.59	0.65	**0.78**	0.70	0.72	0.65
Curved/mixed path	RMS H.A.E/(°)	10.46	**9.56**	23.92	13.49	11.07	23.34
	RMS C.T.E/m	**3.35**	3.81	3.55	3.78	3.90	5.28
	Average reward	0.62	0.66	0.73	0.73	**0.74**	0.69
All paths	RMS H.A.E/(°)	17.46	17.47	**14.83**	14.91	17.39	21.56
	RMS C.T.E/m	3.66	3.50	**3.24**	3.79	4.32	5.27
	Average reward	0.6	0.65	**0.76**	0.71	0.73	0.67

To verify the path dataset's contribution to control performance, 1, 10, 100, and 1 000 random path samples are intercepted from a random path dataset to form a training set. With the other training parameters unchanged, the LSTM-DPPO controller network is trained from scratch and tested with the standard path dataset. The results are given in Table 3. It is found that an increase in path training sample size improves the error and average reward of the controller. Moreover, the table reveals that if the training sample size is 100, the controller has a better performance index on the curved/mixed path, but a poorer performance index on the polygonal path when compared with the size of 1 000. It is assumed to be that controller adjusts the learning policy in the process of self-learning to cope with various paths. When the control policy is completely fitted with the polygonal path, it will sacrifice a certain degree of fitting with the curved/mixed path. Hence, when the volume of learning samples increases, the controller could gradually learn how to balance the control policy through reinforcement learning while taking into account different paths to improve the comprehensive control performance constantly.

Table 3 Comparison results of the RMS for H.A.E. and C.T.E. in the varis scale of data set

Performance indices for standard path datasets		one sample	10 samples	100 samples	1 000 samples
Polygonal path	RMS H.A.E/(°)	21.28	21.60	19.05	**11.19**
	RMS C.T.E/m	4.85	4.37	4.36	**3.12**
	Average reward	0.69	0.654	0.74	**0.78**
Curved/mixed path	RMS H.A.E/(°)	13.23	9.09	**8.70**	23.92
	RMS C.T.E/m	4.90	3.80	**2.95**	3.55
	Average reward	0.69	0.71	**0.77**	0.73
All paths	RMS H.A.E/(°)	18.98	18.02	16.09	**14.83**
	RMS C.T.E/m	4.87	4.21	3.96	**3.24**
	Average reward	0.69	0.67	0.75	**0.76**

6. Conclusions

This paper proposes a path following control method for USV with LSTM-DPPO based deep reinforcement learning. To enhance the controller's generalization and practicality, a path dataset is constructed, and the corresponding LSTM-DPPO algorithm is developed to meet the requirements of model training. The simulation results show that the proposed LSTM-DPPO method could improve the performance of the path following, lower the heading angle and cross-track errors, and increase the average reward along with the increase of training epochs, which is superior in the most cases to other control methods including MLP-PPO and PID. Additionally, the contribution of the path dataset to control performance is also verified in the test. This reveals that having diversity in the training environment for reinforcement learning could be beneficial to the controller's network generalization performance. Future work will focus on the application of the LSTM-DPPO method in the USV in real.

References

[1] LI Z F, LIU Z, ZHANG J Q. Multi-under-actuated unmanned surface vessel coordinated path tracking. Sensors, 2020, 20(3): 864.

[2] WANG N, KARIMI H R. Successive waypoints tracking of an underactuated surface vehicle. IEEE Trans. on Industrial Informatics, 2019, 16(2): 898–908.

[3] HINZE M, SCHMIDT A, LEINE R I. The direct method of Lyapunov for nonlinear dynamical systems with fractional damping. Nonlinear Dynamics, 2020, 102: 2017–2037.

[4] LYU C X, YU H S, CHI J R, et al. A hybrid coordination controller for speed and heading control of underactuated unmanned surface vehicles system. Ocean Engineering, 2019, 176: 222–230.

[5] ROSARIO R V C, CUNHA J P V S. Experimental variable structure trajectory tracking control of a surface vessel with a motion capture system. Proc. of the 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017: 2864–2869.

[6] TEMEL T, ASHRAFIUON H. Sliding-mode speed controller for tracking of underactuated surface vessels with extended Kalman filter. Electronics Letters, 2015, 51(6): 467–469.

[7] FARAMIN M, GOUDARZI R H, MALEKI A. Track-keeping observer-based robust adaptive control of an unmanned surface vessel by applying a 4-DOF maneuvering model. Ocean Engineering, 2019, 183: 11–23.

[8] MA Y L, HAN J D, HE Y Q. Design of strait-line tracking controller of under-actuated USV based on back-stepping method and feedback compensation. Applied Mechanics and Materials, 2011, 48: 391–396.

[9] WANG C X, XIE S R, CHEN H Z, et al. A decoupling controller by hierarchical backstepping method for straight-line tracking of unmanned surface vehicle. Systems Science & Control Engineering, 2019, 7(1): 379–388.

[10] HUANG H B, GONG M, ZHUANG Y F, et al. A new guidance law for trajectory tracking of an underactuated unmanned surface vehicle with parameter perturbations. Ocean Engineering, 2019, 175: 217–222.

[11] WANG N, SUN Z, YIN J C, et al. Fuzzy unknown observer-based robust adaptive path following control of underactuated surface vehicles subject to multiple unknowns. Ocean Engineering, 2019, 176: 57–64.

[12] SHARMA S K, SUTTON R, MOTWANI A, et al. Non-linear control algorithms for an unmanned surface vehicle. Proceedings of the Institution of mechanical engineers, Part M: Journal of Engineering for the Maritime Environment, 2014, 228(2): 146–155.

[13] SUN X J, WANG G F, FAN Y S, et al. A formation

autonomous navigation system for unmanned surface vehicles with distributed control strategy. IEEE Trans. on Intelligent Transportation Systems, 2020, 22(5): 2834–2845.
[14] MARTINSEN A B, LEKKAS A M, GROS S, et al. Reinforcement learning-based tracking control of usvs in varying operational conditions. Frontiers in Robotics and AI, 2020, 7: 32.
[15] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of go without human knowledge. Nature, 2017, 550(7676): 354–359.
[16] VINYALS O, BABUSCHKIN I, CZARNECKI W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 2019, 575(7782): 350–354.
[17] WAN K F , LI B, GAO X G, et al. A learning-based flexible autonomous motion control method for UAV in dynamic unknown environments. Journal of Systems Engineering and Electronics, 2021, 32(6): 1490–1508.
[18] SONG W P , CHEN Z Q, SUN M W, et al. Reinforcement learning based parameter optimization of active disturbance rejection control for autonomous underwater vehicle. Journal of Systems Engineering and Electronics, 2022, 33(1): 170–179.
[19] MAGALHÃES J, DAMAS B, LOBO V. Reinforcement learning: the application to autonomous biomimetic underwater vehicles control. IOP Conference Series: Earth and Environmental Science, 2018, 172(1): 012019.
[20] HUO Y J, LI Y P, FENG X S. Model-free recurrent reinforcement learning for AUV horizontal control. IOP Conference Series: Materials Science and Engineering, 2018, 428(1): 012063.
[21] XU H W, WANG N, ZHAO H, et al. Deep reinforcement learning-based path planning of underactuated surface vessels. Cyber-Physical Systems, 2019, 5(1): 1–17.
[22] ZHAO Y J, QI X, MA Y, et al. Path following optimization for an underactuated USV using smoothly-convergent deep reinforcement learning. IEEE Trans. on Intelligent Transportation Systems, 2020, 22(10): 6208–6220.
[23] WANG N, ZHANG Y, AHN C K, et al. Autonomous pilot of unmanned surface vehicles: bridging path planning and tracking. IEEE Trans. on Vehicular Technology, 2021, 71(3): 2358–2374.
[24] GONZALEZ G A, BARRAGAN A D, COLLADO G I, et al. Adaptive dynamic programming and deep reinforcement learning for the control of an unmanned surface vehicle: experimental results. Control Engineering Practice, 2021, 111: 104807.
[25] LEKKAS A M, FOSSEN T I. Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization. IEEE Trans. on Control Systems Technology, 2014, 22(6): 2287–2301.
[26] NELSON D R, BARBER D B, MCLAIN T W, et al. Vector field path following for miniature air vehicles. IEEE Trans. on Robotics, 2007, 23(3): 519–529.
[27] BIBULI M, BRUZZONE G, CACCIA M, et al. Path-following algorithms and experiments for an unmanned surface vehicle. Journal of Field Robotics, 2009, 26(8): 669–688.
[28] FOSSEN T I, BREIVIK M, SKJETNE R. Line-of-sight path following of underactuated marine craft. IFAC proceedings volumes, 2003, 36(21): 211–216.
[29] WOO J, YU C, KIM N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Engineering, 2019, 183: 155–166.
[30] SUTTON R S, BARTO A G. Reinforcement learning: an introduction. Cambridge: MIT Press, 2018.
[31] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533.
[32] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning. Proc. of the 33rd International Conference on International Conference on Machine Learning, 2016: 1928−1937.
[33] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning. https://arxiv.53yu.com/abs/1509.02971.
[34] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms. https://arxiv.53yu.com/abs/1707.06347.
[35] BØHN E, COATES E M, MOE S, et al. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. Proc. of the International Conference on Unmanned Aircraft Systems , 2019: 523−533.
[36] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust region policy optimization. Proc. of the 32nd International Conference on International Conference on Machine Learning, 2015: 1889−1897.
[37] SCHULMAN J, MORITZ P, LEVINE S, et al. High-dimensional continuous control using generalized advantage estimation. https://arxiv.53yu.com/abs/1506.02438.
[38] HEESS N, TB D, SRIRAM S, et al. Emergence of locomotion behaviours in rich environments. https://arxiv.53yu.com/abs/1707.02286.

Biographies

XIA Jiawei was born in 1994. He received his B.S. and M.S. degrees from Naval University of Engineering (NUE), Wuhan, China, in 2016 and 2019, respectively. He is pursuing his Ph.D. degree in system engineering at NUE, Wuhan, China. His research interests include intelligent control of unmanned surface vehicles, deep reinforcement learning, and multi-agent control.
E-mail: 491650471@qq.com

ZHU Xufang was born in 1978. She received her B.S., M.S., and Ph.D. degrees from Naval University of Engineering (NUE), Wuhan, China, in 1999, 2007, and 2017 respectively. She is a lecturer in the School of Electronic Engineering, NUE. Her research interests include target characteristics, information perception technology, and circuits and systems.
E-mail: 1580284687@qq.com

LIU Zhong was born in 1963. He received his M.S. and Ph.D degrees from Naval University of Engineering (NUE) and Huazhong University of Science and Technology, Wuhan, China, in 1984 and 2004 respectively. He is currently a full professor at Shipborne Commseand & Control Department, NUE. His research interests include nonlinear control of mechanical systems with applcations to robotics and unmanned surface vehicles.
E-mail: liuzh531@163.com

XIA Qingtao was born in 1978. He received his B.S. and M.S. degrees from Naval University of Engineering (NUE), Wuhan, China, in 2001 and 2012, respectively. He is a lecturer in the School of Weaponry Engineering, NUE. His research interests include unmanned system war and command and control system.
E-mail: xiaqing777@163.com