To cite this article: ZHU K, HUANG Z, WANG X M. Tracking control of intelligent ship based on deep reinforcement learning [J/OL]. Chinese Journal of Ship Research, 2020, 16(1). http://www.ship-research.com/EN/Y2020/V16/I1/105.

DOI: 10.19693/j.issn.1673-3185.01940

Tracking control of intelligent ship based on deep reinforcement learning

ZHU Kang[1], *HUANG Zhen*[*1], *WANG Xuming*[2]

1 School of Automation, Wuhan University of Technology, Wuhan 430070, China

2 Intelligent Transport System Research Center, Wuhan University of Technology, Wuhan 430063, China

Abstract: [**Objectives**] The tracking control of intelligent ships often faces the problem of low controller stability in complex control environments and manual algorithmic computing. In order to achieve precise tracking control, this paper proposes a controller based on deep reinforcement learning (DRL). [**Methods**] Guided by the line-of-sight (LOS) algorithm and based on the maneuvering characteristics and control requirements of ships, this paper formulates a path of Markov decision processes by following the control problem, and designs its state space, action space, and reward by applying a deep deterministic policy gradient (DDPG) algorithm to implement the controller. An off-line learning method is used to train the controller. After the training, a comparison is made with BP-PID control to analyze the control effects. [**Results**] Simulation results show that the deep reinforcement learning (DRL) controller can rapidly converge from the training process to meet the control requirements, with the advantages of small yaw error, and a visible reduction in the frequency of changes of the rudder angle. [**Conclusions**] The study results can provide a reference for the tracking control of intelligent ships.

Keywords: intelligent ships; tracking control; deep reinforcement learning (DRL); line-of-sight algorithm

CLC number: U664.82

0 Introduction

Currently, as the research on vehicles is developing in the intelligent and unmanned direction, intelligent ships have attracted much attention from the global shipbuilding and shipping industries. To achieve intelligent and autonomous development, intelligent ships deeply integrate traditional ship design and manufacturing with modern information communication and artificial intelligence, including intelligent navigation, intelligent ship equipment, and intelligent ship testing[1]. Among them, intelligent navigation has been an important foundation of ships to perform cargo transportation, communication and rescue, and other tasks. To make ships comply with normal navigation orders and perform tasks safely and effectively even under many complex water disturbances, it is especially important for us to use effective control means to carry out accurate trajectory tracking.

The studies on trajectory tracking can be divided into guidance and control. In terms of guidance, the line-of-sight (LOS) algorithm is usually used to transform the path tracking problem into a convenient dynamic error control problem. In terms of control, based on the complex non-linear systems of ships, model-free control methods, such as the proportional integral derivative (PID) algorithm, and the model linearization method are usually used to improve the computational rate of nonlinear models. However, in complex environments, traditional PID controllers not only have complex parameters

Received: 2020-04-29 **Accepted:** 2020-07-06

Supported by: National Key Research and Development Program of China (2018YFB1601500)

Authors: ZHU Kang , female, born in 1998, master degree candidate. Research interest: intelligent control of ships and its applications. E-mail: zk13972793427@163.com

HUANG Zhen, female, born in 1974, Ph.D., professor. Research interest: intelligent control theory and its applications. E-mail: h-zhen@163.com

WANG Xuming, male, born in 1964, Ph.D., professor. Research interest: ship intelligence. E-mail: ted@whut.edu.cn

***Corresponding author:** HUANG Zhen

but also do not have adaptive learning capability. However, the control algorithms such as the optimal control algorithm and the feedback linearization algorithm can obtain high control accuracy only by building precise models. Although the sliding mode control does not require high model accuracy, its buffeting problem is difficult to be solved[2]. For some adaptive parameter regulation methods, such as the adaptive PID control method that realizes self-setting of PID parameters by estimating system outputs, there are deviations between system outputs and real outputs due to model uncertainties and external disturbances[3]. Some other adaptive parameter regulation methods consume much time in optimizing parameters, which affects the real-time performance of control. Adaptive PID controllers have a fast response and good real-time performance in the combination with fuzzy logic [4], but their control accuracy depends on complex fuzzy rule bases, which makes the overall calculation complicated.

Considering complex nonlinear system models of ships as well as a large amount of parameter setting and complex computation for ensuring the real-time performance of tracking control, this paper will use the deep reinforcement learning (DRL) algorithm to study the trajectory tracking problem of intelligent ships. DRL is the combination of deep learning and reinforcement learning. It obtains optimization objectives through reinforcement learning and environment exploration, and deep learning provides the operation mechanism to characterize and solve problems. Without relying on dynamic or environmental models, DRL algorithms do not require massive algorithmic computation and can learn by themselves. Based on reinforcement learning, Magalhães et al. [5] designed a supervised switcher using Q-learning and applied it to unmanned surface vehicles (USVs) and the switcher could intelligently switch controllers so that the driving state of USVs could meet different environmental and maneuvering requirements. To improve the stability of complex reinforcement learning, Mnih et al. [6] proposed the deep Q network (DQN) algorithm in 2015 by combining reinforcement learning with deep natural networks. The proposition of this algorithm represents the arrival of the era of DRL. Then this algorithm was applied to navigation collision avoidance of underdriven unmanned ships [7].

Aiming at massive parameter setting, complex algorithm calculation, and other problems, this paper will design a DRL tracking controller based on the deep deterministic policy gradient (DDPG) algorithm to achieve accurate tracking control of ships. Based on the LOS guidance algorithm, this controller controls ship courses to track trajectories. The path tracking problem of ships is modeled as a Markov decision process (MDP) according to maneuvering characteristics and control requirements of actual ships. Then we design the corresponding state space, operation space, and reward function, and use the off-line learning method to train the controller. At last, we use simulation tests to verify the effectiveness of the DRL trajectory controller and compare the control effect with that of the BP-PID controller.

1 Overall design of tracking control system of intelligent ships

1.1 Guidance of LOS algorithm

The tracking control system consists of guidance and control parts, and the guidance part generally works by determining the required course angle according to the trajectory information and the current state of ships. The LOS algorithm used in this paper has been widely used for path control. The LOS algorithm can be combined with controllers under uncertain model parameters and complex operation environments, thus tracking and controlling models. The navigation principle of the LOS algorithm is to generate the desired course based on a variable radius and minimum circles generated near waypoints, namely the LOS angle. Then appropriate control is made so that the current course of ships is the same as the LOS angle, and thus the trajectory tracking can be achieved [8].

The schematic diagram of the LOS algorithm is shown in Fig. 1. It is assumed that the current tracking waypoint is $P_{k+1}(x_{k+1}, y_{k+1})$ and the last waypoint is $P_k(x_k, y_k)$. The radius R_{Los} is selected with $P_s(x_s, y_s)$, the position of the ship, as the center of the circle so that it can intersect with the path P_kP_{k+1}. The point $P_{\mathrm{Los}}(x_{\mathrm{Los}}, y_{\mathrm{Los}})$ near P_{k+1} is selected as the LOS point, and the intersection angle ψ_{Los} between x_0 and the direction vector from the current coordinate of the ship to the LOS point is the LOS angle that should be tracked. In the figure, d is the minimum distance from the current position of the ship to the tracking path, and ψ is the current course angle.

The calculation formulas of the radius R_{Los} are given by Equations (1) and (2). To avoid a zero value of $R_{\min}$, we use two times the ship length L_{pp} for

calculation[9].

$$\begin{cases} a(t) = \sqrt{(x(t)-x_k)^2+(y(t)-y_k)^2} \\ b(t) = \sqrt{(x_{k+1}-x(t))^2+(y(t)-y_{k+1})^2} \\ c(t) = \sqrt{(x_{k+1}-x_k)^2+y_{k+1}-y_k)^2} \\ R_{\min}(t) = \sqrt{a(t)^2-\left(\dfrac{c(t)^2-b(t)^2+a(t)^2}{2c(t)}\right)^2} \end{cases} \quad (1)$$

$$R_{\text{Los}} = R_{\min}(t) + 2L_{\text{pp}} \quad (2)$$

where the calculated $R_{\min}$ is the trajectory error ε at the current moment t, namely d in Fig. 1.

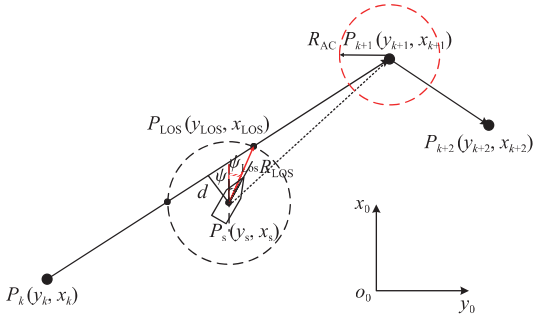


Fig. 1 Schematic diagram of LOS algorithm

When the ship tracks along the path and arrives at a position in a certain neighborhood of the next course point, namely an acceptance circle with $P_{k+2}(x_{k+2}, y_{k+2})$ as the center of the circle and R_{AC} as the radius, the ship updates the current course point as the next one. The radius R_{AC} is generally set as two times the ship length.

1.2 Design of control process based on reinforcement learning

Reinforcement learning (RL), like deep learning, belongs to machine learning and is an important branch of machine learning. It is mainly used to solve continuous decision problems and is an important method for solving MDP problems [10].

The studied problem can be transformed to an MDP problem through models. It can be simply indicated as a quadruple < $\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{P}$, R >, where $\boldsymbol{S}$ indicates the set of all state values, namely the state space; $\boldsymbol{A}$ denotes the action space of the set of action values; $\boldsymbol{P}$ indicates the state transition probability matrix, namely that if the action value $\boldsymbol{A}_t = \boldsymbol{a}$ is selected under the state $\boldsymbol{S}_t = \boldsymbol{s}$ at the moment t, the probability of generating the state $\boldsymbol{s}_1$ at the moment $t+1$ is $P^a_{ss_1} = \boldsymbol{P}[\boldsymbol{S}_{t+1} = \boldsymbol{s}_1 | \boldsymbol{S}_t = \boldsymbol{s}, \boldsymbol{A}_t = \boldsymbol{a}]$; R=r($\boldsymbol{s}$, $\boldsymbol{a}$) is the reward function, which is used to evaluate the action value $\boldsymbol{a}$ under the state $\boldsymbol{s}$. The control part in the tracking control system is indicated by an MDP model, as shown in Fig. 2.

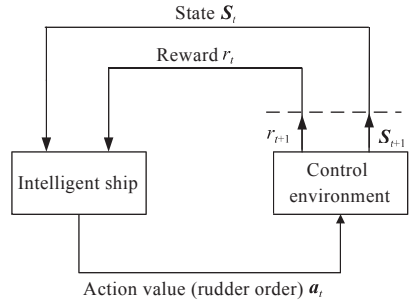


Fig. 2 MDP model of ship control

As shown in Fig. 2, the intelligent ship directly interacts with the current control environment and does not need to acquire any information in advance. During training, the ship uses the action value $\boldsymbol{a}_t$ to interact with the environment, thus updating the state from s_t to s_{t+1} and acquiring the corresponding reward r_{t+1}. Then, the ship continues to take the next action to interact with the environment. In this process, massive data are generated, which are used to optimize the policy π for selecting its actions. Simply, it is a cyclic iterative process. In reinforcement learning, the training objective is to find an optimal control policy π* to maximize the accumulative reward R_t [11]. In the below formula, γ denotes the discount factor, which is used to measure the value proportion of the future reward at the current stage. It is set that $\gamma \in [0, 1]$.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=1}^{\infty} \gamma^k r_{t+k+1} \quad (3)$$

Policy π can be evaluated by two value functions, the state-value function $V^{\pi}(\boldsymbol{s}_t)$ and the action-value function $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. $V^{\pi}(\boldsymbol{s}_t)$ is the expectation function of the accumulative reward under the state of following the current policy, and E denotes the expected value. Similarly, $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the expectation function of the accumulative reward under the specific state and action ($\boldsymbol{s}_t$, $\boldsymbol{a}_t$).

$$V^{\pi}(\boldsymbol{s}_t) = E_{\pi}[R_t | \boldsymbol{s}_t] = E_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1} | \boldsymbol{s}_t\right] \quad (4)$$

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = E_{\pi}[R_t | \boldsymbol{s}_t, \boldsymbol{a}_t] = E_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right] \quad (5)$$

According to the value functions and the definition of the optimal control policy π*, policy π* always follows the following condition:

$$\pi^* = \arg\max V^{\pi}(\boldsymbol{s}_t) = \arg\max Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) \quad (6)$$

1.3 Markov modeling of the trajectory tracking problem

From the above, it can be seen that the component design in the Markov modeling process is the most critical in the reinforcement learning-based

control design. The correctness of the state space, action space, and reward has a great impact on the algorithm performance and the convergence speed. Thus, we carry out the Markov modeling design for the trajectory tracking problem of intelligent ships.

1) Design of state space.

According to the LOS algorithm used for guidance, we adjust the current course angle according to the LOS angle to achieve the tracking effect. Thus, when selecting the state, we should take the output parameters in the LOS algorithm into consideration, including the difference e between the objective course ψ_{LOS} and the actual course ψ, the trajectory error ε and the distance error ε^{d} with trajectory points.

The ship model can acquire the surging velocity u, the swaying velocity v, the yawing velocity r of the bow, and the rudder angle δ. To make reinforcement learning realize high-accuracy tracking and rapidly suit the transformation of many different environments, we add the state value at the last moment as well as the error $e(k-1)$ between the current course error and the course error at the last moment for comparison in addition to the state value at the current moment. Thus, the current state can better indicate whether the ship moves towards the direction with smaller errors. At last, the state space of the current moment t can be designed as

$$\begin{aligned}\boldsymbol{s}_t = [&e_t,\ \varepsilon_t,\ \varepsilon_t^{\mathrm{d}},\ u_t,\ v_t,\ r_t,\ \delta_t,\ e(k-1)_t,\\ &e_{t-1},\ \varepsilon_{t-1},\ \varepsilon_{t-1}^{\mathrm{d}},\ u_{t-1},\ v_{t-1},\ r_{t-1},\ \delta_{t-1}]\end{aligned} \tag{7}$$

2) Design of action space.

In light of the characteristics of the trajectory tracking task and the principle of the LOS guidance algorithm, this paper focuses on the control of the navigation direction of ships, namely the rudder angle, but does not consider the control of the ship speed and the propeller speed. The action space only includes one action value, the rudder order, which is denoted by δ. It should be set according to the control requirements of ships, which are set in the range (−35°, 35°). The maximum rudder speed is 15.8 (°)/s.

3) Design of reward function.

The reward increases as the expected course angle approximates the LOS angle or as the error between the actual trajectory and the objective trajectory reduces. Thus, the reward function is designed as a piecewise function, which is a general form of it.

$$r_t = \begin{cases} 0, & \text{if } |e| \leqslant 0.1 \text{ rad} \\ -|e| - 0.1\,|e(k-1)| - 0.01\,|\varepsilon|, & \text{if } |e| > 0.1 \text{ rad} \end{cases} \tag{8}$$

where $e(k-1)$ indicates the difference between the current course error and the course error at the last moment. When the difference is bigger than 0.1 rad, the reward is set as negative, which also can be called the penalty value. By doing this, we hope the training network can change the current bad state rapidly. The action under the negative reward is compared with that under a zero reward at another segment. Thus, the controller can rapidly select actions with higher rewards after training and learning, thus reaching the optimal effect.

1.4 Overall scheme of the control system

The overall framework of the reinforcement learning-based trajectory control system of intelligent ships is shown in Fig. 3. The LOS algorithm calculates the required course and the trajectory error based on the current position of the ship and then integrates them with the state information of the ship to form the above state vector $\boldsymbol{s}_t$ which is then input to the trajectory controller. After that, according to the reinforcement learning algorithm, we output the current optimal action value $\boldsymbol{a}_t$ to operate the ship, and calculate the corresponding reward by the reward function r_t to iterate the controller parameters, so that the trajectory controller can learn by itself.

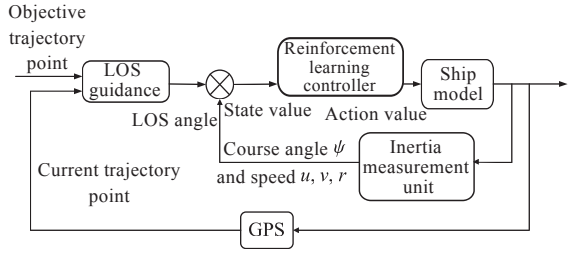


Fig. 3 Block diagram of tracking control of intelligent ships based on reinforcement learning

Before putting the controller into real-time control, we first need to train the controller offline. After the training with certain times is set, the obtained network parameters that maximize the accumulative reward are stored and integrated so that the reinforcement learning controller can be obtained. It is then applied to the real-time control system for trajectory tracking.

There are many algorithms, mechanisms, and network structures for solving reinforcement learning problems, but these methods are not expandable and can only deal with low-dimensional problems. Thus, Mnih et al. [6] proposed a training method, the DQN algorithm, that can use large-scale neural net-

works in reinforcement learning problems. This algorithm successfully combines deep learning with reinforcement learning, allowing reinforcement learning to be extended to deal with some decision problems in high-dimensional states and action spaces [12]. The DQN algorithm can solve the problem of unstable or even divergent learning results due to the mutual interference between the reinforcement learning and the training of the neutral network approximator for the approximation of the value function [13]. Thus, this algorithm is a pioneer in the DRL field.

The DQN algorithm significantly improves the stability and performance of complex reinforcement learning problems. However, since it uses discrete action spaces, it should discretize the output actions and can only select the optimal action from limited action values. For the trajectory tracking problem of ships, it is difficult to control intelligent ships accurately if there are too few candidate actions. To make the algorithm satisfy the maneuvering characteristics and requirements of ships, this paper uses a DRL algorithm applicable to the continuous action space, namely the DDPG-based algorithm [14], to design the tracking controller of intelligent ships. This algorithm can not only operate on the continuous action space but also process a large amount of data efficiently and accurately.

2 Controller design based on DDPG algorithm

2.1 Principle of DDPG algorithm

Lillicrap et al. [14] applied the DQN algorithm to continuous actions and thus proposed an Actor-Critic model-free algorithm based on a deterministic policy gradient. The basic framework of DDPG is shown in Fig. 4.

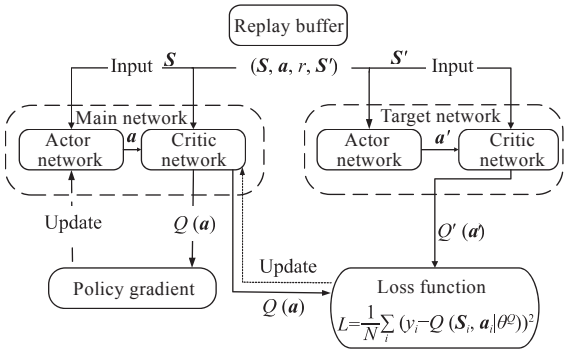


Fig. 4 Block diagram of DDPG

The overall network adopts the Actor-Critic form, which includes both the neural networks based on the value function and the neural networks based on the policy gradient. θ^{π} in the Actor network indicates the deterministic policy function $\boldsymbol{a}=\pi(\boldsymbol{s}|\theta^{\pi})$, and θ^{Q} in the Critic network denotes the value function $Q(\boldsymbol{s},\boldsymbol{a}|\theta^{Q})$. In addition, by referring to the DQN technology, DDPG eliminates the instability brought by large-scale neural networks by using the experience replay mechanism and a single target network.

The experience replay means that the current state, action, and other information are stored at each moment as the experience $\boldsymbol{e}_t=(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ of the intelligent body, thus forming a replay memory buffer $\boldsymbol{D}=\{e_1, \ldots, e_N\}$. When training the network, we randomly extract a mini-batch amount of experience data from it as training samples. However, the operation of reusing historical data will increase the data usage and also disrupt the sequence of the original data, which will reduce the correlation between the data. The target network builds two neutral networks with the same structure, the main network for updating the parameters of the neural network and the target network for generating the optimal target values. In the beginning, the parameters of the main network are given to the target network, and then the parameters of the main network are continuously updated while the target network remains unchanged. After a period of time, the parameters of the main network are given to the target network again. This cyclic operation can make the optimal target values stable over a period of time, thus making the algorithm performance more stable.

In the training process, the Actor network in the main network selects the optimal action value $\boldsymbol{a}$ through the current policy function $\boldsymbol{a}=\pi(s|\theta^{\pi})$ and gives it to the intelligent ship according to the sample state $\boldsymbol{s}$ randomly selected from the experience pool. The state value $\boldsymbol{s}'$ at the next moment can be obtained after the intelligent ship interacts with the environment. Meanwhile, the Critic network accepts the current state $\boldsymbol{s}$ and the action value $\boldsymbol{a}$, and uses the value function $Q(\boldsymbol{s}, \boldsymbol{a}|\theta^{Q})$ to evaluate the expected accumulative reward of the current state, which is used to update the parameters of the Actor network. The overall target network receives the state $\boldsymbol{s}'$ at the next moment. The action is selected by the target Actor network and given to the target Critic network, so that the target expectation $Q'(\boldsymbol{a}')$ can be obtained. Then, we update the parameters of the Critic network of the main network by calculating the loss function. For the parameter updating of

the Actor network in the main network, Silver et al. [15] found that the gradient of the objective function $J(\theta^{\pi})$ of the deterministic policy using the policy π is equivalent to the expected gradient of the Q function using the policy π:

$$\frac{\partial J(\theta^{\pi})}{\partial \theta^{\pi}} = E_{\mathrm{s}}\left[\frac{\partial Q(\boldsymbol{s},\boldsymbol{a}|\theta^{Q})}{\partial \theta^{\pi}}\right] \tag{9}$$

According to the deterministic policy $\boldsymbol{a}=\pi(s|\theta^{\pi})$, the gradient of the Actor network can be obtained as

$$\frac{\partial J(\theta^{\pi})}{\partial \theta^{\pi}} = E_{\mathrm{s}}\left[\frac{\partial Q(\boldsymbol{s},\boldsymbol{a}|\theta^{Q})}{\partial \boldsymbol{a}}\frac{\partial \pi(\boldsymbol{s}|\theta^{\pi})}{\partial \theta^{\pi}}\right] \tag{10}$$

$$\nabla_{\theta^{\pi}} J \approx \frac{1}{N}\sum_{i}(\nabla_{\boldsymbol{a}} Q(\boldsymbol{s},\boldsymbol{a}|\theta^{\pi})|_{\boldsymbol{s}=\boldsymbol{s}_i,\boldsymbol{a}=\pi(\boldsymbol{s}_i)} \cdot \nabla_{\theta^{\pi}}\pi(\boldsymbol{s}|\theta^{\pi})|_{\boldsymbol{s}=\boldsymbol{s}_i}) \tag{11}$$

On the other hand, the value gradient of the Critic network is

$$\frac{\partial L(\theta^{Q})}{\partial \theta^{Q}} = E_{\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}'\sim D}\left[(\mathrm{Target}Q - Q(\boldsymbol{s},\boldsymbol{a}|\theta^{Q}))\frac{\partial Q(\boldsymbol{s},\boldsymbol{a}|\theta^{Q})}{\partial \theta^{Q}}\right] \tag{12}$$

$$\mathrm{Target}Q = r + \gamma Q'(\boldsymbol{s}', \pi(\boldsymbol{s}'|\theta^{\pi'})|\theta^{Q'}) \tag{13}$$

where $\theta^{\pi'}$ and $\theta^{Q'}$ indicate the parameters of the target policy network and the target value function network. The updating method of the target network is different from that in the DQN algorithm. In the DDPG algorithm, the parameters of the Actor-Critic network's target networks are updated gradually, which is also called soft updating. In this way, the stability in the learning process can be further improved:

$$\theta^{Q'} = \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{14}$$

$$\theta^{\pi'} = \tau\theta^{\pi} + (1-\tau)\theta^{\pi'} \tag{15}$$

where τ is the learning rate.

We define the minimum loss function to update the parameters of the Critic network, where y_i is the error between the evaluation value function of the state and action at the current moment and the target expectation obtained by the target network:

$$L = \frac{1}{N}\sum_{i}(y_i - Q(\boldsymbol{s}_i,\boldsymbol{a}_i|\theta^{Q}))^2 \tag{16}$$

2.2 Steps of algorithm

The parameters of the Actor-Critic network are initialized, and the parameters of the current networks are given to the corresponding target networks. We initialize the experience pool by setting the capacity of the replay buffer as 30 000, the soft updating learning rate as 0.01, and the accumulative discount factor as 0.9. The steps of the training are as follows.

1) Initialize the ship environment;

2) Repeat the following steps until the set maximum step is reached;

3) In the main network, the Actor network obtains the state $\boldsymbol{s}_t$ of the ship at the current moment, selects the action rudder order δ_{t} according to the current policy and gives it to the ship for operating, namely $\delta_t = \pi(\boldsymbol{s}_t|\theta^{\pi})$;

4) After carrying out the current rudder order, the ship outputs the reward r_t and the next state s_{t+1}. The Actor network acquires this state information and selects the next rudder order δ_{t+1};

5) The data $(\boldsymbol{s}_t, \delta_t, r_t, \boldsymbol{s}_{t+1})$ generated in this process is stored in the replay buffer as the data set for training networks. When the experience pool is full, the data is cyclically stored from the first location;

6) N samples $(\boldsymbol{s}_t, \delta_t, r_t, \boldsymbol{s}_{t+1})$ are randomly selected from the replay buffer as the training data of the current Actor network and the Critic network;

7) The Critic network is updated through the loss function, and the current Actor network is updated through the policy gradient update of the Actor network. Then we conduct soft updating for the target networks.

3 Simulation and comparison of algorithms

3.1 Establishment of the simulation environment

To verify the above method, we simulate the ship tracking in the Python environment. The single-propeller single-rudder KVLCC2 tanker with a length of 7 m introduced in References [16-17] is used as the model of the research object. The model with three degrees of freedom (surging, swaying, and yawing) is used for modeling, and the modeling details can refer to Reference [16]. The major parameters of the ship are listed in Table 1.

Table 1 Parameters of a KVLCC2 tanker

Parameter	Value	Parameter	Value
Ship length L_{pp}/m	7	Block coefficient C_{b}	0.809 8
Ship width B_{wl}/m	1.168 8	Coordinate of buoyant center/m	0.244 0
Moulded depth D/m	0.656 3	Diameter of propeller D_{p}/m	0.216 0
Volume of displacement/m^3	3.272 4	Area of rudder /m^2	0.053 9

For the DDPG controller in this paper, the parameters of the Critic network and the Actor network are respectively set as Tables 2 and 3.

3.2 Off-line learning of controller

The DDPG-based off-line learning is set as fol-

Table 2 Parameters of Critic network

Parameter	Value
Input layer	State vector $\boldsymbol{S}(t)$
First hidden layer	300
Activation function at the first layer	Relu
Second hidden layer	200
Activation function at the second layer	Relu
Output layer	Action $\delta(t)$
Activation function at the output layer	Tanh
Initialization of parameters	Initialization of Xavier
Learning rate	0.000 1
Optimizer	Adam

Table 3 Parameters of Actor network

Parameter	Value
Input layer	State vector $\boldsymbol{S}(t)$, action $\delta(t)$
First hidden layer	300
Activation function at the first layer	Relu
Second hidden layer	200
Activation function at the second layer	Relu
Output layer	$Q(\boldsymbol{S}(i),\delta(i))$
Activation function at the output layer	Linear
Initialization of parameters	Initialization of Xavier
Learning rate	0.001
Optimizer	Adam

lows: initialize the network parameters and the experience buffer pool; design the maximum number of training rounds of 2 000, the maximum step of each round of 500, and the sampling time of 1 s. When we plan the trajectories to be tracked during the training, we design multiple three-waypoint routes based on the corner transformation by referring to the design principle stated in Reference [18], and we randomly select one route for trajectory tracking in each training round. The purpose is to adapt the controller to multiple environments and meet the requirements of the LOS guidance algorithm on the course control.

During training, the data is stored in the experience pool, and then a set of data is randomly selected for training. The states and action values are normalized. When the maximum step is reached or the final waypoint is output, the round is stopped and the total reward of this round is calculated. The course errors of 200 rounds, 300 rounds, and 500 rounds are shown in Fig. 5. It can be seen from the figure that as the training rounds increase, the course error decreases significantly and the control algorithm keeps converging. The training stops when it reaches the maximum rounds, and the total reward is continuously increasing. To make the image display more clearly, we cut out the total rewards of 200 to 500 rounds, which are depicted in Fig. 6. It can be seen that the algorithm basically converges at about 270 rounds, which indicates the fast learning process.

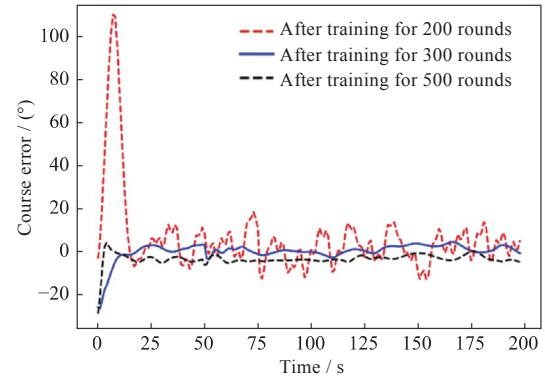


Fig. 5 Curves of course error

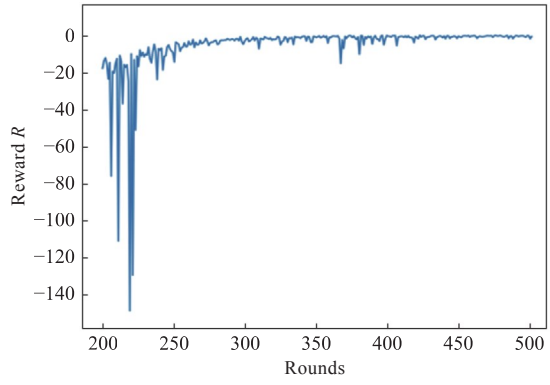


Fig. 6 Curve of total reward

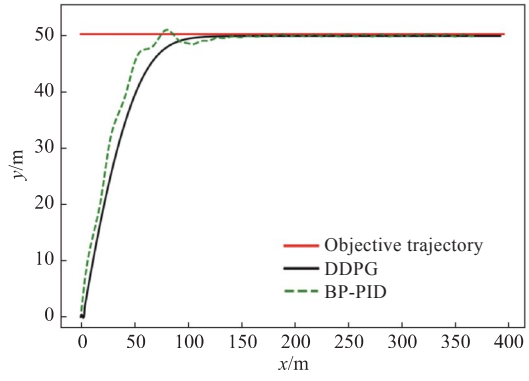
3.3 Design and comparison of simulation tests

When the above training is completed, the DDPG controller saves the network parameters with the largest reward function and applies them to the trajectory tracking simulation. To verify the feasibility of the DDPG controller, we select the BP-PID controller for comparative analysis.

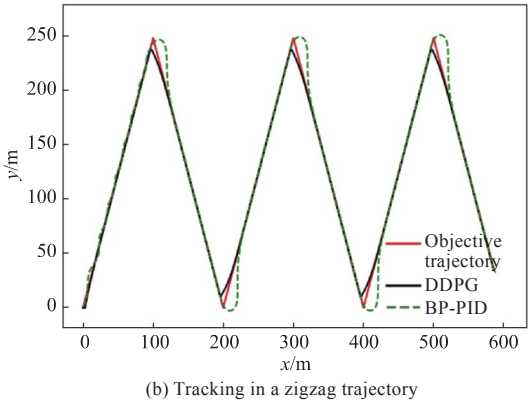
In terms of the BP-PID controller for comparison, we use the BP neural network with four nodes at the input layer, five nodes at the hidden layer, and three nodes at the output layer to select three parameters of the PID. The learning rate is 0.546 and the factor of momentum is 0.79. By referring to Reference [19], we use additional inertia terms to optimize the neural network. The DDPG controller and the BP-PID controller are simulated and com-

pared in the same environment. In the simulation, the ship starts from the origin (0, 0) with an initial course of 45°, an initial speed, namely the surging speed u of 1.179 m/s, and an initial propeller speed r of 10.4 r/s.

Experiment 1: We design the straight trajectory and the zigzag trajectory to observe the tracking effect of the two controllers for straight routes and routes with sharp changes (see Fig. 7). The coordinates of the trajectory points are respectively (0, 50), (400, 50) and (0, 0), (100, 250), (200, 0), (300, 250), (400, 0), (500, 250), (600, 0) with the unit of m.



(a) Tracking in a sgraight trajectory



(b) Tracking in a zigzag trajectory

Fig. 7 Trajectory tracking result (Experiment 1)

The comparison of the two types of trajectory tracking shows that the DDPG controller can stably track the straight trajectory more rapidly, and is significantly better than the BP-PID controller in tracking the zigzag trajectory. The root-mean-square error (RMSE) of the course angle is calculated (see Fig. 7(b)), which is 61.017 8 in the BP-PID controller and it is only 10.018 in the DDPG controller. This indicates that the latter has a much better control performance.

Experiment 2: To simulate the trajectory of traditional ships, we design the trajectory with waypoints of (0, 0), (100, 50), (150, 250), (400, 250), (450, 50), (550, 0) for tracking. The tracking curves and the RMSEs of the course of the two controllers are compared, as shown in Fig. 8 and Table 4, respectively.

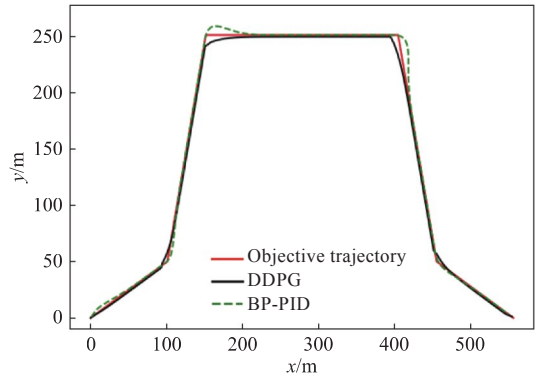
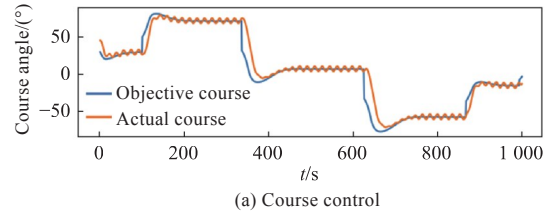


Fig. 8 Trajectory tracking result (Experiment 2)

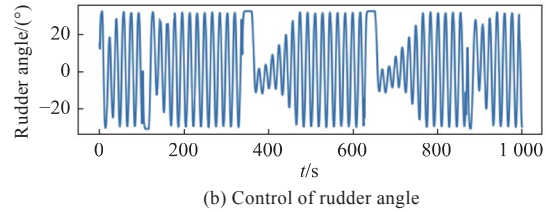
Table 4 Control performance

Controller	RMSE
BP-PID controller	13.585 0
DDPG controller	6.911 96

In this simulation, the tracking effects of the two controllers for the LOS angle and the changing frequency of the rudder angle are further compared, as shown in Figs. 9 and 10, respectively. The total cruise time of the PID controller is about 1 000 s after the parameter setting of the BP neural network, while the cruise time of the DDPG controller is reduced by 4%. In the course tracking at corners, the DDPG controller can achieve the desired value within 20 s, and the adjustment time of the BP-PID controller is about 60 s. In addition, the control effect of the BP-PID controller is not stable and the rudder angle has a high vibration frequency. It can be seen that the DRL controller can quickly make adjustments according to the trajectory changes. By reducing unnecessary control links, it has less adjustment time. Moreover, with a stable control effect and small change frequency of the rudder an-

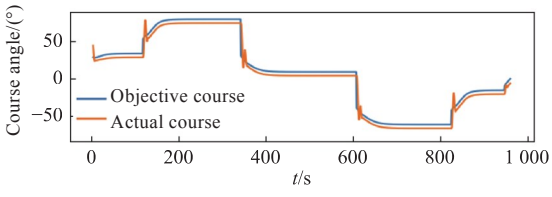


(a) Course control

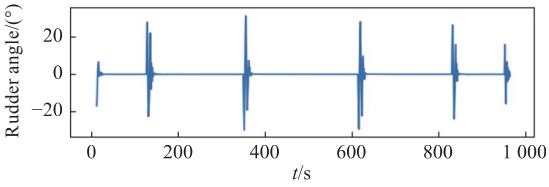


(b) Control of rudder angle

Fig. 9 Control result of BP-PID



(a) Course control



(b) Control of rudder angle

Fig. 10 Control result of DDPG

gle, the DDPG controller has better control performance.

4 Conclusion

Aiming at the trajectory tracking problem of ships, this paper proposed a tracking controller based on DRL. First, on the basis of the LOS guidance algorithm, we built a Markov model for tracking control and provided the program of the algorithm based on the DDPG controller. Then we developed simulation tests for the tracking control system in the Python environment and compared it with the BP-PID controller.

The trajectory tracking problem was first modeled into an MDP problem, and then the controller was trained offline. The analysis of this process reveals that the DDPG controller can converge quickly to the control requirements in training. This verifies the feasibility of the designed states, the action space, and the reward function. Moreover, the comparison results of the trajectory tracking simulation tests show that the DDPG controller can respond to the trajectory changes quickly. With a stable control effect and smaller changes in the rudder angle, it can well adapt to different trajectories. Generally, the control method based on DRL can be applied to the tracking control of ships. With adaptive and stable control capability, this method not only avoids complicated control calculation but also ensures real-time performance. This study has a certain reference value for the intelligent control of ships.

References

[1] YAN X P, LIU J L, FAN A L, et al. Development and trend of intelligent ship technology [J]. Ship Engineering, 2020, 42 (3): 15−20 (in Chinese).

[2] GUO B ZH. An introduction to active disturbance rejection control for nonlinear systems [J]. Mathematical Modeling and its Applications, 2017, 6 (1): 13−22, 52 (in Chinese).

[3] ZHANG X W, XIE L, CHU X M, et al. An overview of path following control methods for unmanned surface vehicles [J]. Journal of Transport Information and Safety, 2020, 38 (1): 20−26 (in Chinese).

[4] LIU S, XING B W, ZHU W L. A fusion fuzzy PID controller with real-time implementation on a ship course control system [C] //Proceedings of the 2015 23rd Mediterranean Conference on Control and Automation (MED). Torremolinos, Spain: IEEE, 2015.

[5] MAGALHÃES J, DAMAS B, LOBO V. Reinforcement learning: the application to autonomous biomimetic underwater vehicles control [J]. IOP Conference Series: Earth and Environmental Science, 2018, 172: 12−19.

[6] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529−533.

[7] WOO J, KIM N. Vector field based guidance method for docking of an unmanned surface vehicle [C] //Proceedings of the 12th ISOPE Pacific/Asia Offshore Mechanics Symposium. Gold Coast, Australia: International Society of Offshore and Polar Engineers, 2016.

[8] HAN P, LIU ZH L, ZHOU Z C, et al. Path tracking control algorithm based on LOS method for surface self-propulsion vessel [J]. Applied Science and Technology, 2018, 45 (3): 66−70 (in Chinese).

[9] MOREIRA L, FOSSEN T I, SOARES C G. Path following control system for a tanker ship model [J]. Ocean Engineering, 2007, 34 (14/15): 2074−2085.

[10] REN Y, ZHAO SH T. Deep reinforcement learning based path following control of magnetic navigation AGV [J]. Journal of Hangzhou Dianzi University, 2019, 39 (2): 28−34 (in Chinese).

[11] CARRERAS M, RIDAO P, EL-FAKDI A. Semi-online neural Q_leaming for real-time robot learning [C] //Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems. Las Vegas, Nevada: IEEE, 2003: 662−667.

[12] LIU J W, GAO F, LUO X L. Survey of deep reinforcement learning based on value function and policy gradient [J]. Chinese Journal of Computers, 2019, 42 (6): 1406−1438 (in Chinese).

[13] WOO J, YU C, KIM N. Deep reinforcement learning based controller for path following of an unmanned surface vehicle [J]. Ocean Engineering, 2019, 183: 155−166.

[14] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning [C] // Proceedings of the 4th International Conference on Learning Representations. San Juan, 2015: A187.

[15] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms [C] //Proceedings of the 31st International Conference on Machine Learning. Beijing, China: ACM, 2014: I-387−I-395.

[*Continued on page* 60]

基于改进的视线导引算法与自抗扰航向控制器的无人艇航迹控制

杨忠凯[1], 仲伟波[*1,2], 冯友兵[1], 孙彬[1]
1 江苏科技大学 电子信息学院, 江苏 镇江 212003
2 江苏科技大学 海洋装备研究院, 江苏 镇江 212003

摘　要: [**目的**] 无人艇(USV)在复杂环境情况下会出现偏离目标航线的情况, 为提高水面无人艇的抗干扰能力及实际航行的稳定性, 实现对航迹的准确控制, 提出一种改进的无人艇航迹控制方法。[**方法**] 根据导航信号受环境影响的情况, 对GPS信号有效和无效2种情况下的航迹控制分别进行分析, 在自主可控平台上设计并实现了基于模糊控制可变船长比的视线导引算法(LOS)和自抗扰航向控制器(ADRC)相结合的航迹控制方法, 并开展了双桨双舵无人艇湖上试验。[**结果**] 仿真结果表明: 该方法可满足航迹控制的要求, 转弯后航向能够快速保持稳定, 无频繁摆舵现象, 且该方法能够完成真实环境下的航迹控制, 航迹贴线误差均值约为0.1 m, 方差约为0.03。[**结论**] 湖上试验结果验证了该算法在实际工程应用中的可行性和有效性。

关键词: 航迹控制; 视线导引算法; 自抗扰算法; 模糊控制; 自主可控平台

[*Continued from page* 51]

[16] YASUKAWA H, YOSHIMURA Y. Introduction of MMG standard method for ship maneuvering predictions [J]. Journal of Marine Science and Technology, 2015, 20 (1): 37–52.

[17] LIU J L, QUADVLIEG F, HEKKENBERG R. Impacts of the rudder profile on manoeuvring performance of ships [J]. Ocean Engineering, 2016, 124: 226–240.

[18] WANG Y. Modeling and path tracking control of unmanned surface vessel [D]. Hangzhou: Zhejiang University, 2019 (in Chinese).

[19] ZHONG H X, QIU S H, LUO X SH, et al. Study of applying BP neural network with inertia term self-tuning to attitude stability of quadrotor unmanned aerial vehicle [J]. Journal of Guangxi Normal University (Natural Science Edition), 2017, 35 (2): 24–31 (in Chinese).

基于深度强化学习的智能船舶航迹跟踪控制

祝亢[1], 黄珍[*1], 王绪明[2]
1 武汉理工大学 自动化学院, 湖北 武汉 430070
2 武汉理工大学 智能交通系统研究中心, 湖北 武汉 430063

摘　要: [**目的**] 智能船舶的航迹跟踪控制问题往往面临着控制环境复杂、控制器稳定性不高以及大量的算法计算等问题。为实现对航迹跟踪的精准控制, 提出一种引入深度强化学习技术的航向控制器。[**方法**] 首先, 结合视线(LOS)算法制导, 以船舶的操纵特性和控制要求为基础, 将航迹跟踪问题建模成马尔可夫决策过程, 设计其状态空间、动作空间、奖励函数; 然后, 使用深度确定性策略梯度(DDPG)算法作为控制器的实现, 采用离线学习方法对控制器进行训练; 最后, 将训练完成的控制器与BP-PID控制器进行对比研究, 分析控制效果。[**结果**] 仿真结果表明, 设计的深度强化学习控制器可以从训练学习过程中快速收敛达到控制要求, 训练后的网络与BP-PID控制器相比跟踪迅速, 具有偏航误差小、舵角变化频率小等优点。[**结论**] 研究成果可为智能船舶航迹跟踪控制提供参考。

关键词: 智能船舶; 航迹跟踪控制; 深度强化学习; 视线导航法