Ocean Engineering 243 (2022) 110166

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Obstacle avoidance strategy for an autonomous surface vessel based on modified deep deterministic policy gradient

Chang Zhou, Yiting Wang *, Lei Wang, Huacheng He

State Key Laboratory of Ocean Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
SJTU-Sanya Yazhou Bay Institute of Deepsea Science and Technology, Sanya 572024, China

ARTICLE INFO

Keywords:
Obstacle avoidance
ASV
Reinforcement learning
MDDPG

ABSTRACT

In the present paper, a decision-making agent based on reinforcement learning is designed for establishing an obstacle avoidance strategy of an autonomous surface vessel (ASV). To solve the sparse feedback in obstacle avoidance issue, the Modified Deep Deterministic Policy Gradient (MDDPG) method is proposed in the present study, using an actor-network to generate actions from the state of the ASV and a critic-network to evaluate the behavior of the actor-network. The gradient descend method is applied to improve the networks. Memory pool modification, success pool modification and target network modification are developed to smooth the training process of the ASV. A comparative analysis based on numerical simulations are carried out using Deep Q Network (DQN) method, Modified Deep Q Network (MDQN) method and MDDPG method. The results demonstrate that the MDDPG method smooths and accelerates the learning process. Compared with the other two methods, the effectiveness and viability of the proposed method is significantly increased. Random obstacles tests are also conducted to confirm the applicability and generalization of the method.

1. Introduction

Obstacle avoidance is one of the most fundamental capabilities for an autonomous surface vessel (ASV) based on the consideration of safety. Various obstacle avoidance strategies have been studied and developed by many researchers. The algorithms can be roughly divided into geometric method, optimization method, virtual vector method and artificial intelligence method (Zhang et al., 2021). Typical applications of geometric method are visibility graph algorithm (Lozano-Pérez and Wesley, 1979), voronoi diagram algorithm (Voronoi, 1908) and free space algorithm (Lozano-Perez et al., 1984). For optimization method, dynamic window algorithm (Fox et al., 1997), Dijkstra algorithm (Dijkstra et al., 1959), A* algorithm (Doran and Michie, 1966) and ant colony algorithm (Goss et al., 1989) show ideal capability. The virtual vector methods like vector potential algorithm (Masoud and Masoud, 2000) work for some certain cases. These approaches can achieve the objectives of avoiding obstacles under specific assumptions. Although these conventional methods can find feasible solutions, they may lead to heavy computation loads due to the complexity of the environment. These methods may give unreliable solutions with incomplete perception of global obstacles. In addition, the path generated by the mathematical methods may not be directly applied as the trajectory because of the dynamic constraints of the underactuated ASV. Nowadays, the generalization ability of the artificial intelligence methods, especially reinforcement learning (RL) algorithms, offers a feasible and efficient alternative to solve the computation loads explosion. Once the obstacle avoidance strategy model is established and well-trained, a few computations are required to generate the appropriate actions for the ASV.

Applications of RL for marine vehicle controlling have been widely carried out these years. The similar researches can be categorized into three types: ASV navigation maneuvering, collision avoidance decision-making and obstacle avoidance control. For ASV maneuvering problems, the RL algorithm is expected to drive the ASV to follow a given path or a designed control sequence. Łącki (2008) implemented the algorithms like V-learning, Q-learning, and State–Action–Reward–State–Action (SARSA) to examine the learning performance in ship maneuvering. Martinsen and Lekkas (2018) adopted the RL algorithm to solve the straight-path following problem for underactuated ships by minimizing the cross-track errors. In the case of Woo et al. (2019), the desired course angle was set as the setpoint value for path following, and the strategy was verified and validated by a series of experiments in open water. Yoo and Kim (2016) presented a strategy based on the RL algorithm to optimize the marine vehicle route, which took the disturbance of the ocean environment like wind, wave, and current into consideration.

* Correspondence to: State Key Laboratory of Ocean Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail address: zjgwangyiting@sjtu.edu.cn (Y. Wang).

https://doi.org/10.1016/j.oceaneng.2021.110166
Received 17 April 2021; Received in revised form 21 September 2021; Accepted 5 November 2021
Available online 1 December 2021
0029-8018/© 2021 Elsevier Ltd. All rights reserved.

For collision avoidance decision making, the RL algorithm is adopted to explore and discover the optimal behavior strategy in an unknown environment (Wang et al., 2018). Layek et al. (2017) implemented the Normalized Advantage Function to steer an underactuated ship passing through a gate in numerical simulations. Woo and Kim (2020) further proposed a collision avoidance strategy for ASVs on a given path. The information of ship path, dynamic obstacles, and static obstacles was input into three different neural network layers and was extracted to high-dimensional space by convolutional layers. A physical experiment was carried out to validate the strategy when encountering dynamic obstacles. A knowledge-free path planning approach in a fixed map for intelligent ships was developed by Chen et al. (2019) based on Q-learning. The effectiveness and feasibility of the strategy was confirmed by comparing with other conventional methods. The obstacle locations were given and unchanged during the training process. Cheng and Zhang (2018) gave an obstacle avoidance method based on Q-learning without inputting the beforehand well-planned path. The obstacle information was given when detected by the ASV.

The RL algorithm in obstacles avoidance control only focus on preventing collision. Prior experience is often adopted to determine whether it is necessary to implement the obstacle avoidance behavior. The multi-vessel obstacle avoidance strategy was proposed by Shen et al. (2019), aiming to prevent collisions between ships in restricted water. Ship domain, bumper, and safe area are designed by experience to guarantee the safety. The navigation mode of the ship will be switched between "normal" and "obstacle avoidance" according to the ship bumper. Zhao et al. (2019) proposed an RL algorithm in compliance with International Regulations for Preventing Collisions at Sea (COLREGS) for preventing ASV from collisions with encountered obstacles. The ASV is designed to follow the predefined path when coming across the obstacles. The multiship collision case was further studied, improved and validated by Zhao and Roh (2019).

In most cases (Xu et al., 2020; Woo and Kim, 2020; Chen et al., 2019; Cheng and Zhang, 2018), the ASV is fully-censored and fully informed of the position and state of itself, the position and scale of all the obstacles as well as the position of the target point. However, the information is sparse in the real world, and the ASV may be informed of a few nearby blocks. The reference route may not be generated or given when the environmental information is not fully detected. Therefore, an RL strategy based on incomplete perceptual information urges to be developed. When implementing the Deep Q Network (DQN) to the decision-making controller, the output of the DQN (so-called action space) is normally discrete and only a few rudder angles of the ASV can be selected (Mnih et al., 2015). For practical vessel maneuvering, it is of significance to develop a network with non-discrete outputs. In addition, compared to conventional RL applications, the obstacle avoidance is faced with sparse rewards problem, making the updating of the model unstable. The model, therefore, cannot be well trained at the beginning and will be improved quite slowly.

To overcome the limitations discussed above, the present paper proposes a modified deep deterministic policy gradient strategy based on the RL method. The algorithm is proposed based on the following design criteria. First, the ASVs are under an incomplete perception environment. The ASV can only be informed of the obstacles within the detecting distance. The ASV has a limited vision of the obstacles and there is occlusion between obstacles. Second, the actions can arbitrarily be selected by the ASV within the allowable range. Third, The obstacle avoidance algorithm should be universal. The well-trained RL model has a good performance not only in the training environment but also in other environment with random obstacles. Finally, the algorithm has capability of avoiding possible collisions as well as sailing to the target point. In addition, a reward function is designed and adopted to realize the obstacle avoidance under incomplete environment information. An actor-network and a critic-network are applied to generate and modify the continuous rudder angles. Target network modification, memory pool modification, and success pool modification are developed and implemented to accelerate the model training and avoid the cold launch. Numerical simulations including static obstacle avoidance, dynamic obstacle avoidance and random obstacles avoidance have been performed to confirm the feasibility and reliability of the algorithm.

The main contributions of present paper are summarized as:

1. An incomplete perception simulation environment has been established. Corresponding sensors, determiners, decision-makers, and ship dynamics models are designed.
2. The Modified Deep Deterministic Policy Gradient (MDDPG) method is developed as the ASV's decision-making controller under an incomplete perception environment. With the MDDPG method, continuous control output of the ASV is achieved. Simulations results show that the MDDPG method has a good performance for obstacle avoidance problems.
3. A universal reward function based on the incomplete perception model is designed. The obstacle avoidance and navigation function of the ASV is developed. The applicability and universality of the algorithm have been validated by random obstacle tests and dynamic obstacle tests.
4. Multiple modifications of the DDPG model are applied to improve the capability of the obstacle avoidance strategy. The model training is accelerated and the cold launch issue in the RL method is well improved.

The rest of the paper is organized as follows: In Section 2, a virtual environment model is built and the ship dynamics model is established; Section 3 describes collision avoidance decision making strategy based on RL algorithm; the memory pool modification method, success pool modification method, and target network modification method are then introduced in detail; results and discussions are presented in Section 4; Section 5 gives the concluding remarks on present paper.

2. Obstacle avoidance modeling of an underactuated ASV with incomplete perception

2.1. Mathematical design of the environment and sensors

A virtual environment with D in length and width is designed and established based on Python for the ASV navigation and obstacle avoidance with incomplete perception. The safety circles of obstacles with different radii are adopted for vessels to avoid the collision. The number of obstacles in the environment is controlled within a density, denoted as ρ_{obs}. When generating a numerical environment, obstacles are randomly produced and the obstacle density ρ_t is calculated by

$$\rho_t = \frac{\sum_i \pi R_i^2}{D^2} \tag{1}$$

If $\rho_t > \rho_{obs}$, the random generation of the obstacles will be stopped. Here, the ρ_{obs} is set as 0.01.

In the present study, the sensors of the ASV are assumed as a group of rays with different angles. The maximum obstacle-detecting distances of these rays are l. If the rays are blocked by an obstacle, the obstacle will be detected and the distance between the ASV and the obstacle will be acquired by the sensors as l_{dist}. The total number of rays is n. The detection of the target point does not rely on the sensor rays. The location is given directly to the ASV.

Fig. 1 shows the definitions of the established virtual environment in the present paper. The green square represents the target point, the gray solid circles are the safety circles of the obstacles, and the red boat represents the ASV. δ is the angle between the real route and the planned route of the ASV. The blue solid and dotted lines respectively stand for the rays of the sensors detecting the obstacle or not. The normalized feedbacks L_{nor} are adopted to represent the

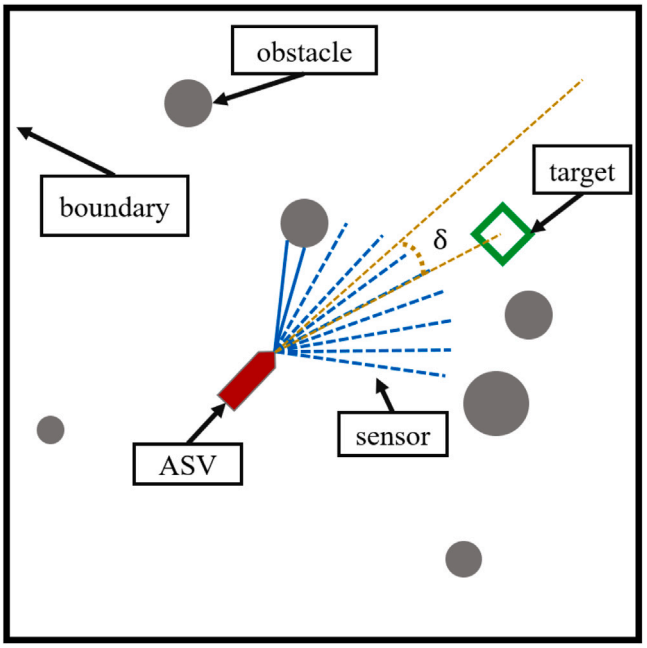


Fig. 1. ASV, sensors and obstacles in the virtual environment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

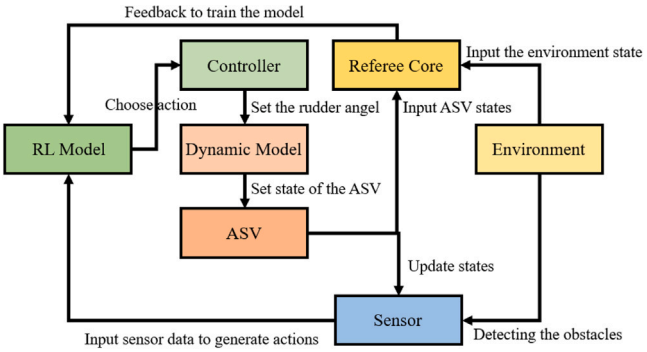


Fig. 2. The framework of the numerical simulation.

sensors detection results. For ith ray, the normalized feedback $l_{nor,i}$ can be expressed as follow:

$$l_{nor,i} = \begin{cases} \frac{l_{dist}}{l}, & \text{Obstacles detected} \\ 1, & \text{Obstacles not detected} \end{cases} \tag{2}$$

Therefore, the L_{nor} can be given by,

$$L_{nor} = [l_{nor,1}, l_{nor,2}, \ldots, l_{nor,n}]^T \tag{3}$$

2.2. Numerical simulation framework

The simulation framework is shown in Fig. 2. The simulation architecture is divided into seven parts. The RL model is responsible for selecting the optimal action based on the state acquired by the sensors. Once the actions are determined, the controller module will execute the corresponding rudder angle and transfer relevant information to the ship dynamics module. Based on the dynamic model, the motion equation of the ASV is solved at each time step. The position, heading, and other information will be updated. The sensor module is responsible for detecting obstacles in the environment. The referee module accesses

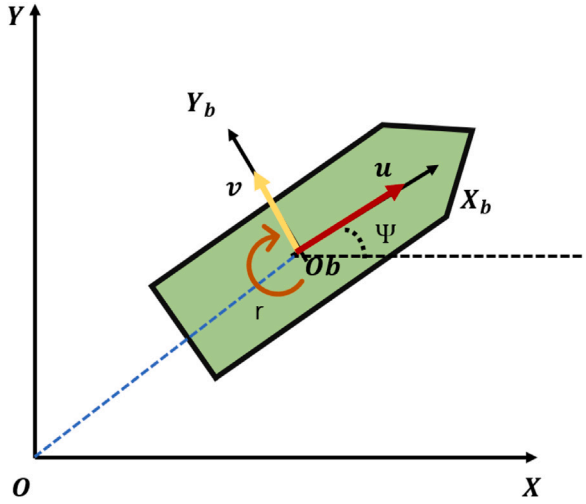


Fig. 3. The coordinate systems of the ASV..

Table 1
Nomenclature of motion equation.

Symbols	Description
ν	Velocity (surge, sway and yaw direction)
η	Displacement (surge, sway and yaw direction)
$M_{RB}\dot{\nu}$	Rrigid-body forces, coriolis–centripetal part
$C_A(\nu_r)\nu$	Hydrodynamic forces, coriolis–centripetal part
$D(\nu_r)\nu$	Hydrodynamic forces, damping part
$M_A\dot{\nu}$	Hydrodynamic forces, added mass part
$g(\eta)$	Rigid-body forces, hydrostatic part
τ	Control inputs
τ_{wind}	Wind forces
τ_{wave}	Wave forces

the state of the ASV and gives the judgment according to the state. The feedback will be returned to the RL module for further training.

2.3. Dynamic model of an underactuated ASV

The coordinate systems are shown in Fig. 3. The coordinate system XOY represents the global coordinate system, and $X_bO_bY_b$ is the rigid body coordinate system. u is the longitudinal speed, and v is the lateral speed. Ψ represents the heading angle. The obstacle avoidance problem is the focus of present paper, and thus, only surge, sway and yaw motions are considered, respectively referring to longitudinal motions, sideways motions, and rotations about the vertical axis of the ASV. The 3 degree of freedom (DOF) maneuvering equation can be written as (Fossen, 2002):

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu + C_A(\nu)\nu + M_A\dot{\nu} + D(\nu)\nu + g(\eta) + g_0 = \tau + \tau_{wind} + \tau_{wave} \tag{4}$$

where the notations of Eq. (4) are listed in Table 1. In this paper, the Cybership II is applied as the research object and the detailed major parameters can be referred to (Skjetne et al., 2004). To emphasize the focus of the present paper, the influence of the wave and wind are ignored. The initial longitudinal velocity u in X_b-axis is set as 1.092 m/s, and the initial acceleration $\dot{\nu}$ is 0 m/s^2.

3. Obstacle avoidance strategy based on the RL method

The decision-making progress used in the present study is first discussed. For comparative studies, both the DQN method and the DDPG methods are designed in this section. To overcome the inherent limitations of the DQN and DDPG method, target network modification, memory pool modification and success pool modification are also developed and presented in this section.

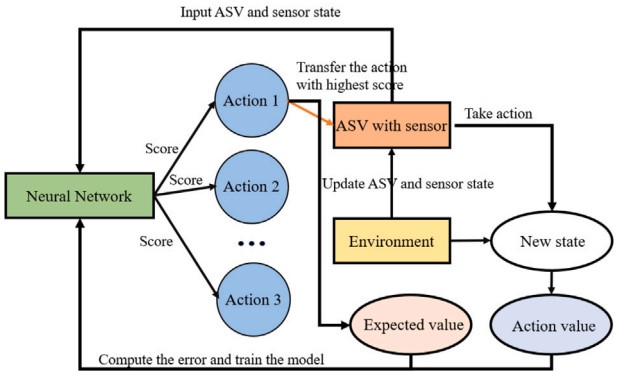


Fig. 4. The framework of the DQN.

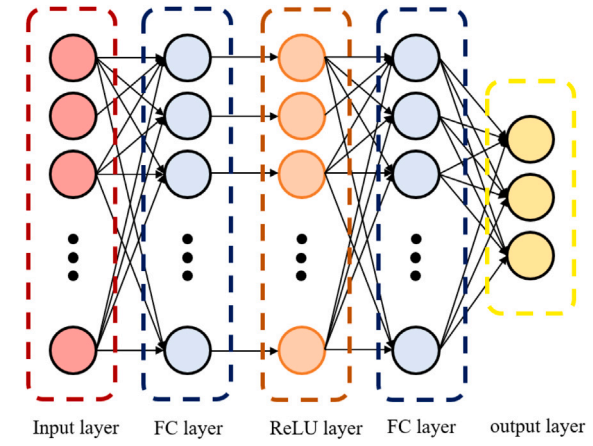


Fig. 5. Structure of the DQN.

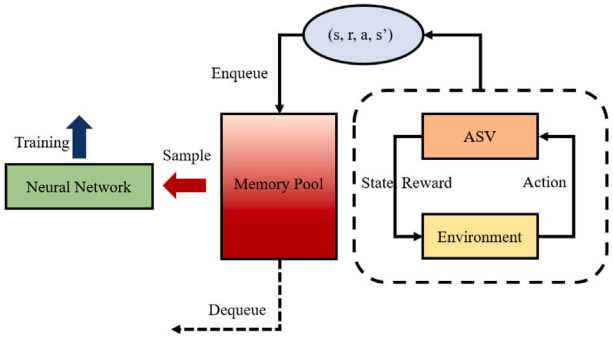


Fig. 6. The framework of the memory pool modification.

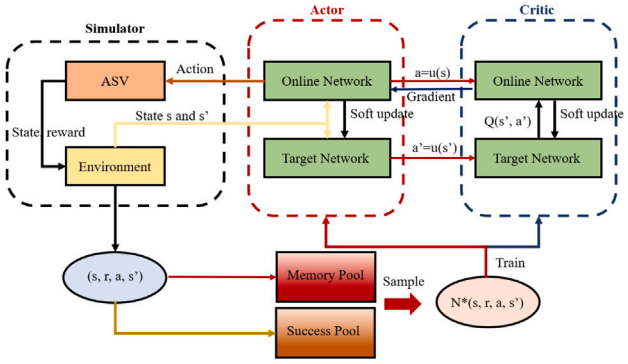


Fig. 7. The framework of the MDDPG method.

3.1. Decision-making progress

The target of the intelligent obstacle avoidance is to find an optimal strategy to prevent collision effectively and efficiently, which can be recorded as $\pi : S \rightarrow A$ (Wang et al., 2018). In the process of obstacle avoidance, there is an expected value for any actions. The optimal strategy is to choose actions based on the state s to maximize the total value gained in the obstacle avoidance process. Once state s is given, the strategy will select and perform a corresponding action a. When the strategy is determined, the behavior of the entire Markov Decision Process (MDP) (Bellman, 1957) can be regarded as a Markov chain:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1}, \ldots, s_{t-1} \xrightarrow{a_{t-1}} s_t \tag{5}$$

If the ASV only chooses the best action considered by the strategy, actions that not so valuable in the perspective of the current strategy will not be performed by the ASV, thus possible better actions will not be tried. The strategy cannot be improved. All actions are required to be made and selected with a certain probability according to the environment. The higher the selected action value is, the larger the action probability will be, which can be expressed as follows:

$$a_t^* = \arg\max_{a_t,i} p\left(a_{t,i} | \left\{s_0, a_0, s_1, \ldots, s_t\right\}\right) \tag{6}$$

where $p\left(a_{t,i} | \left\{s_0, a_0, s_1, \ldots, s_t\right\}\right)$ means the possibility of choosing action $a_{t,i}$ under current strategy in the obstacle avoidance process $\{s_0, a_0, s_1, \ldots, s_t\}$. $a_{t,i}$ denotes the ith action can be chosen under time t.

It is assumed that there is no correlation among the states in the present study. Once the action is performed, the following state is only affected by the dynamic model. The formula of decision-making can be simplified as follows:

$$a_t = \arg\max_{a_i} p\left(a_i | s_t\right) \tag{7}$$

$p(a_i|s_t)$ denotes the possibility of choosing action a_i under state s_t.

3.2. Deep Q Network

In the present paper, we first develop an obstacle avoidance strategy based on the conventional RL method DQN. The value of different actions responding to the environment is approximated by value iteration. The decision-making function is formulated by selecting the most valuable action in the corresponding state of which the actions are limited to several choices. The neural network assess different choices and selects the highest-scoring action. The selected action will be transferred to the controller for further execution. The framework of the DQN is illustrated in Fig. 4.

The update of the Q function can be expressed as follows:

$$q^T(s,a) = q^{T-1}(s,a) + \alpha\left[r\left(s'\right) + \max_{a'} q^{T-1}\left(s', a'\right) - q^{T-1}(s,a)\right] \tag{8}$$

where q^T represents the Tth updated value of q. $r(s')$ is the reward of state s', α the greedy factor, $q(s,a)$ the value of action a under state s.

The neural network is utilized to reduce the computational loads caused by the conventional Q-learning algorithm since each state–action transition is calculated. The inputs of the neural network are based on the ASV and environmental states detected by sensors. The action space is the moment on the rudder, which can be described as follows:

$$a(t) = \begin{cases} \tau_{rudder} \\ 0 \\ -\tau_{rudder} \end{cases} \tag{9}$$

where

$$\tau_{rudder} = [0, 0, M_{moment}]^T \tag{10}$$

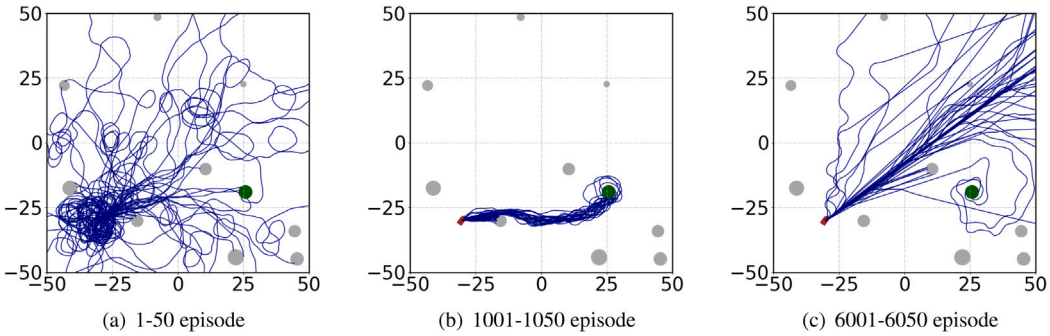


Fig. 8. The obstacle avoidance training progress using DQN method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

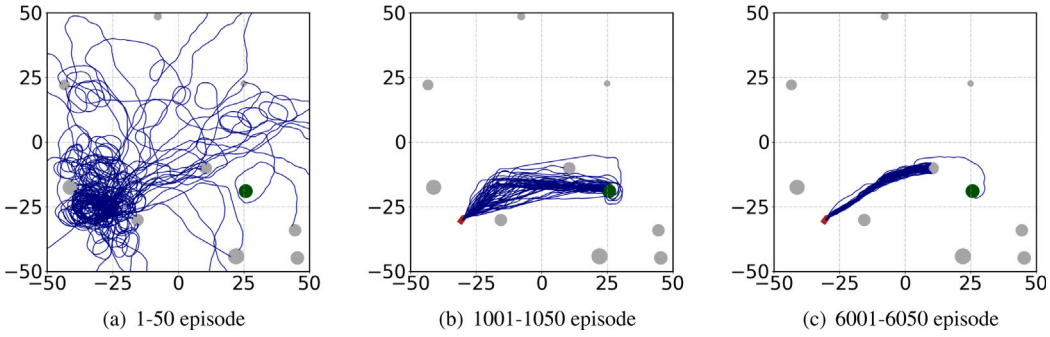


Fig. 9. The obstacle avoidance training progress using MDQN method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the present paper, the DQN is designed with two fully connected (FC) layers and one Rectified Linear Unit (ReLU) layer, as shown in Fig. 5. The dimension of the input layer is 22, including the distance to the target, the angle between the real heading and the target heading of the ASV, longitudinal velocity, normal velocity and angular acceleration of the ASV, and the sensor data. The following layer is a FC layer with an input dimension of 22 and an output dimension of 32. The third layer is a ReLU layer for nonlinear mapping. Subsequently, another FC layer is utilized, of which the input dimension is 32 and output dimension is 3. The output is the score of the three action spaces.

3.3. Deep Deterministic Policy Gradient

In DQN method, since the value of the current action is updated based on the maximum estimated action value of the following state–action transition, the problem of divergence may occur in the continuous exploration space (Sutton and Barto, 2018). In addition, the DQN method determines the action space at first, indicating that the number of optional actions is limited and only a few rudder angles can be selected by the controller of the ASV.

The DQN method is based on value function, and the decision is made by estimating the value of the given action. However, for DDPG method (Lillicrap et al., 2015), the decision-making is implemented based on a strategy function (Silver et al., 2014), which can be written as,

$$a_t = \mu\left(s_t \mid \theta^{\mu}\right) \tag{11}$$

a_t is the action at time t, s_t is the state at t, μ is the strategy function, and θ is the parameter of the strategy function.

The Bellman equation of the action-value function can be described as follows:

$$Q^*(s,a) = \mathrm{E}\left[r(s,a) + \gamma \max_{a'} Q\left(s', a'\right)\right] \tag{12}$$

where $Q^*(s,a)$ is the updated value of choosing action a under state s. $r(s,a)$ is the reward of transition (s,a), which is generated by the reward function. $Q(s',a')$ is the state–action value under transition (s',a') estimated by the network. s' is the state the ASV reached after taking action a under state s.

A mean-squared Bellman error (MSBE) function is developed to calculate the approximation of the state–action value iteratively. The state–action value is generated by a critic-network with the parameters of ϕ to evaluate the score of action a under the state s. D denotes the transition (s,a,r,s').

$$L(\phi, D) = \mathrm{E}\left[\left(Q_\phi(s,a) - \left(r + \gamma \max_{a'} Q_\phi\left(s', a'\right)\right)\right)^2\right] \tag{13}$$

Introducing the actor-network with parameter θ to generate actions under state s, the formula can be written as

$$L(\phi, D) = \mathrm{E}\left[\left(Q_\phi(s,a) - \left(r + \gamma Q_\phi\left(s', \mu_\theta\left(s'\right)\right)\right)\right)^2\right] \tag{14}$$

Where E means the mathematical expectation under all transition set D and strategy ϕ, L denotes the loss function based on MSBE. γ is a constant between 0 and 1. The design of the reward function keeps the same as the one in the DQN method. The structure of the actor-network and critic-network is similar to the DQN. The difference is that, in the actor-network, a tanh-function layer replaces the output layer, which generates the action from 1 to −1 for the ASV. There is one more dimension of the input layer added to the critic-network because the action generated by actor-network is considered. The dimension of the output layer is reduced to 1 to evaluate action.

3.4. Modification of the DQN and DDPG method

3.4.1. Target network modification

During the simulation, the model is updated by the TD-error, which is calculated by the action rewards. Each round of actions may generate some fluctuations during the exploration due to the instability of the

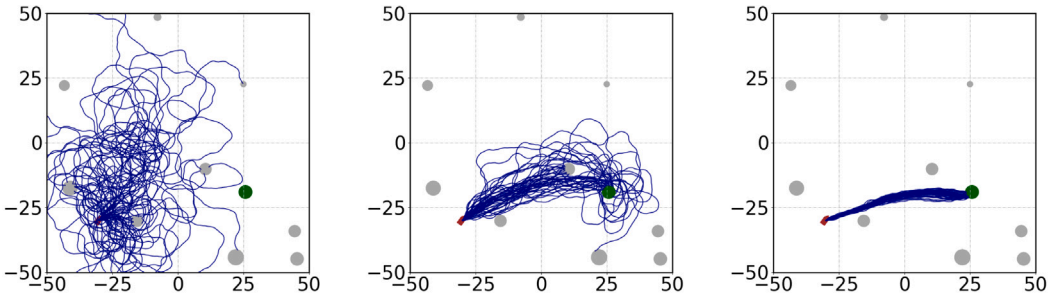


Fig. 10. The obstacle avoidance training progress using MDDPG method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

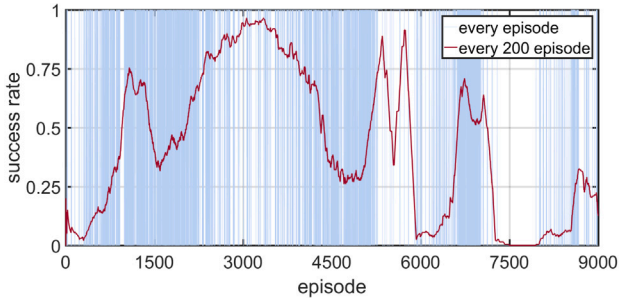


Fig. 11. Success rate in the DQN method.

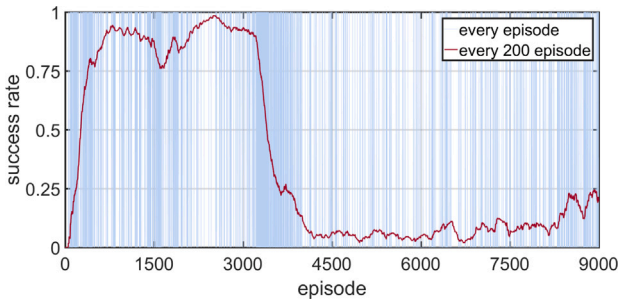


Fig. 12. Success rate in the MDQN method.

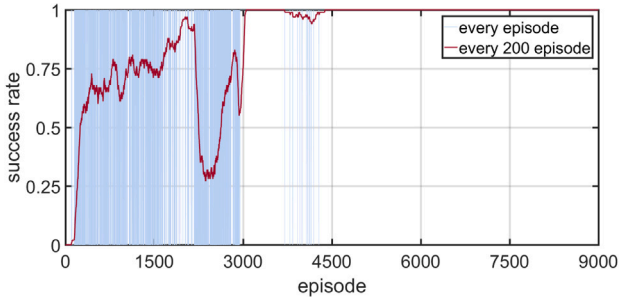


Fig. 13. Success rate in the MDDPG method.

training network. The fluctuation is immediately reflected to the action assessment, making it hard to generate the proper action, and affecting the further estimations.

The network adopted to predict and evaluate the action values simultaneously may lead to divergence. Target network modification is thus developed by establishing a network with the same structure of the original one. These two networks are asynchronously updated to ensure the stability of training. The target network is updated with the updating of the main network by polyak averaging:

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \tag{15}$$

Where ρ is the polyak, a constant between 0 and 1.

3.4.2. Memory pool modification

In this paper, we assume that the obstacle avoidance process for an ASV is a Markov decision-making process. However, from the aspect of practice, the state of the ASV at the future moment will inevitably be affected by the state at current moment. The memory pool modification is applied to maintain the reliability of the model and improve the efficiency of data usage.

All transitions (s, a, r, s') are exported to the real-time updated memory pool. In the training process of the strategy, data are randomly selected to feed the neural network. The framework of present memory pool modification algorithm is shown in Fig. 6.

3.4.3. Success pool modification

In the RL process, the training effect of the neural network is greatly influenced by the data quality. If there is no better action selection data, the training of the neural network will become more difficult to be converged. A success memory pool is developed to restore the state–action transitions of the episodes that the ASV successfully arrives at the target point. In each training, the data are sampled from two sources: the success pool and the memory pool.

The framework of the MDDPG method is depicted as Fig. 7, and the detail of the MDDPG method is given in algorithm 1, 2, and 3 in the appendix.

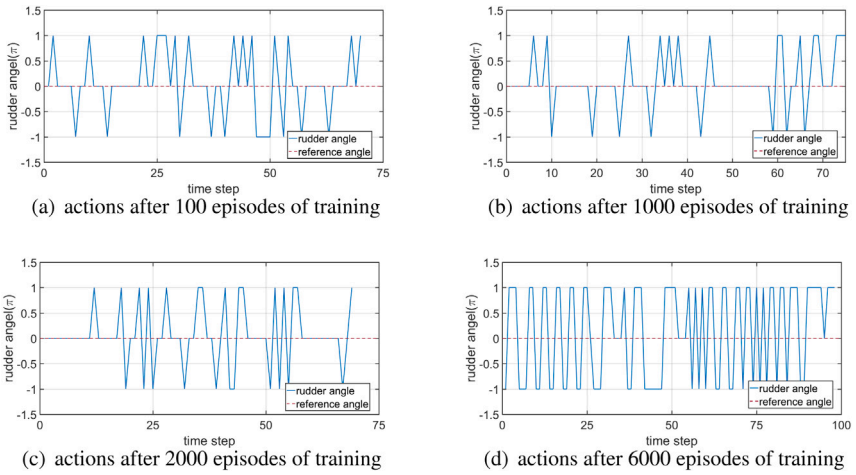
3.5. Reward function design for obstacle avoidance and navigation problem

The reward function is computed as the sum of scores accumulated in each step, where the score is a measurement of action quality by a priori subjective evaluation. The objective of the study is to maneuver the ASVs for obstacle avoidance and voyage navigation.

As can be found in some similar researches (Woo and Kim, 2020; Woo et al., 2019; Shen et al., 2019; Zhao and Roh, 2019; Martinsen and Lekkas, 2018; Wang et al., 2018), the following tasks are usually required to be achieved by the reward functions:

1. Follow the given path.
2. Make collision avoidance decisions.
3. obstacle avoidance control.

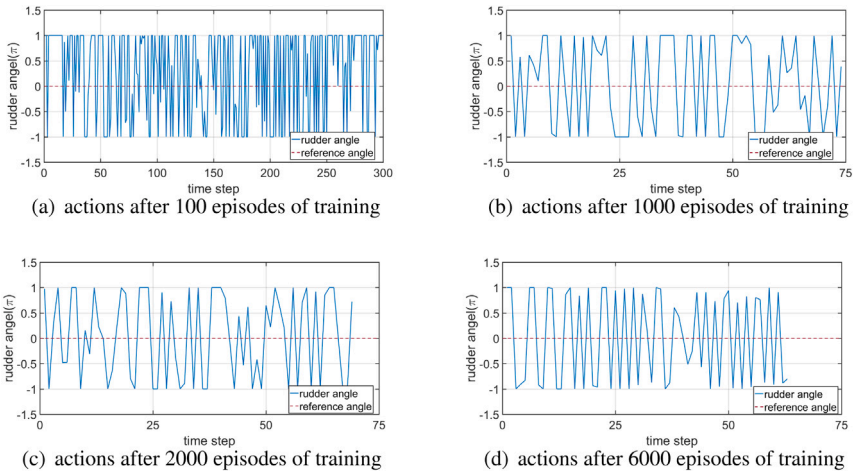
For task type 1, the reward function is only expected to facilitate the path-following. The reward scores generally consist of cross error penalty, track error penalty, and other parts. Cross error penalty and track error penalty help to make the ASV keep as close to the given path as possible (Woo et al., 2019; Martinsen and Lekkas, 2018). For task type 2, the composition of the reward varies depending on the task requirements. In some studies (Woo and Kim, 2020; Wang et al.,



(a) actions after 100 episodes of training (b) actions after 1000 episodes of training

(c) actions after 2000 episodes of training (d) actions after 6000 episodes of training

Fig. 14. Actions taken by MDQN method within different periods.



(a) actions after 100 episodes of training (b) actions after 1000 episodes of training

(c) actions after 2000 episodes of training (d) actions after 6000 episodes of training

Fig. 15. Actions taken by MDDPG method within different periods.

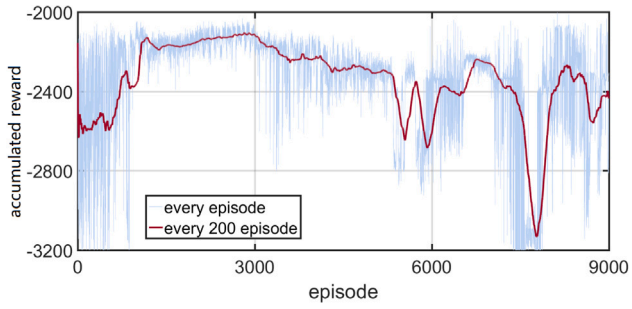


Fig. 16. Accumulated reward based on the DQN method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

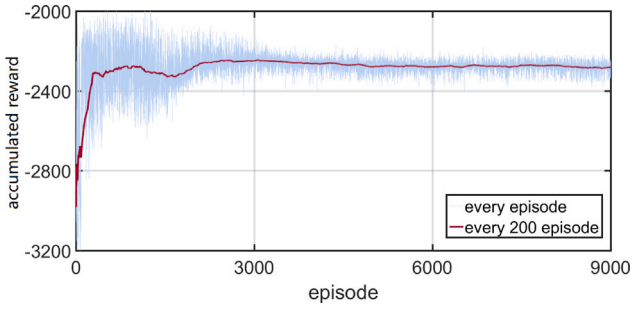


Fig. 17. Accumulated reward based on the MDQN method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2018), the navigation task is taken into consideration by introducing a given path. The reward function can be specified to reward the ASV for approaching its goal and penalize the ASV for collisions and off-courses. In task type 3, the reward function less concentrate on other behaviors (Shen et al., 2019; Zhao and Roh, 2019).

In the present paper, both the obstacle avoidance and navigation problems of the ASV are required to be solved. In task type 2, the navigation problem is mainly solved by giving a well-planned course. In task type 3, the problem is generally settled by switching the ASV to the navigation mode based on mathematical methods. A given path requires a good perception of global obstacles and environments, while in the real world it is difficult to be achieved. On the other hand, switching between navigation mode and obstacle avoidance mode requires prior experience design. Therefore, a reward function design based on incomplete perception and less prior experience is proposed in this paper to give more flexible applications of the present obstacle avoidance strategy.

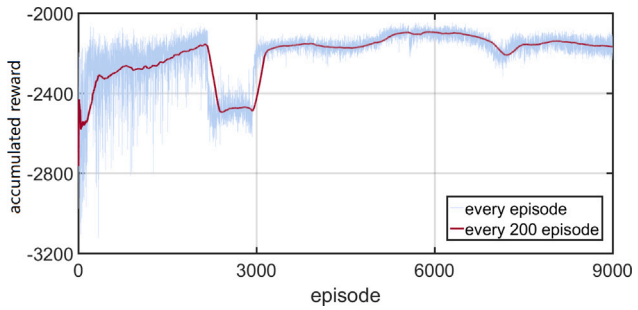


Fig. 18. Accumulated reward based on the MDDPG method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The reward function is developed according to the following assumptions:

1. ASV is designed to learn the right behavior by itself according to the state with a well-designed reward function. A more effective and flexible control strategy than switching control mode with prior experience can be acquired using the RL model.
2. The ASV is under an incomplete perception environment. The environmental information should not be contained in the reward patterns. To strengthen the environment adaption of the RL model, the absolute distance from the ASV to the target point should not be used in the reward function directly since the mean reward value changes with the map size.
3. The ASV learns a good path planning strategy under an incomplete perception environment. Most reward functions solve the navigation problem by following a given path, while in an incomplete perception environment, the course is difficult to be generated. In addition to the obstacle avoidance strategy, the navigation strategy can also be learned by the RL model through the same reward function.
4. The design of the reward function should be as concise as possible. The complex reward function design contains too many prior experience components, while the strategy should be trained by the model itself.

If the ASV arrives at the target point, a positive feedback is given. If a collision occurs, a negative feedback is given. If neither arrival nor collision is achieved, a small score is deducted motivating the ASV to save the sailing time. The offset angle δ between the bow and the target point is introduced as a reference. The larger the offset angle is, the more points will be deducted. The reward function is designed as follows:

$$r = \begin{cases} -C_1 & (\text{if crashed}) \\ -k|\delta| - C_2 & (\text{if not crashed}) \\ C_3 & (\text{if arrived}) \end{cases} \tag{16}$$

where $|\delta|$ is the absolute value of the offset angle. The reward parameters C_1, C_2, C_3 and k are positive numbers. When the ASV collides, a hefty penalty $-C_1$ is given to the model. When the ASV reaches the target point, the model gains a positive reward C_3. If the ASV neither arrives nor collides, the regular penalty $-C_2$ guarantees that the model will be punished continuously and the ASV must promptly reach the target point in order to refrain from the penalty. The angle penalty $-k|\delta|$ makes the ASV sail as straight as possible to the target point.

4. Results and discussion

In the present study, numerical simulations are carried out with three models. The normal DQN algorithm without modifications of the target network method, the memory pool method, and the success pool method is first developed. The Modified DQN (MDQN) is then used as the second control group. The MDDPG method is carried out to be the experimental group. The neural network is adopted in MDDPG to directly output the action of ASV instead of the scores of several actions.

In order to ensure that the RL model can be effectively explored in the early stage, there is a certain probability that some actions of the ASV may not be executed according to the decision made by the RL model. In the DQN method, after an action at each time step is chosen, there will be a certain probability that the final action is randomly selected from the action space. As the training time increases, the probability of disobeying the model selection decreases. The action is taken as the center of the normal distribution, and the final action is selected from it with a probability. The variance of the distribution is gradually reduced with the increase of the training time. The departure to collision or arrival process is regarded as an episode.

The single decision-making time of the DQN and the MDQN method is 0.0002 s while for the MDDPG method it is 0.0004 s. It can be found that the time consumption of the MDQN method equals that of the DQN method, because the modifications are only utilized in the training process. While for the decision-making process, only the action-network is needed. The structures of the two methods' action-networks are the same, leading to the equal time consumption. For the MDDPG method, the action-network structure is changed, causing the change of time consumption. The computational time of these three methods is acceptable for navigation.

4.1. Training process comparison

The simulations using different methods are respectively conducted for at least 9000 training episodes. The neural network training effects are evaluated by comparing different trajectory characteristics in different episodes. Figs. 8–10 demonstrate the moving trajectories of the ASV in different periods applying different RL methods. The gray circles are obstacles, the green circle represents the target position, and the red boat denotes the starting position. The trajectories of ASV are presented by blue lines.

Fig. 8 shows the obstacle avoidance trajectories of an ASV base on the DQN method at different stages. The obstacle avoidance performance of the ASV is relatively poor using the DQN method. With multiple episodes of training, the model shows the overfitting phenomena. When encountering obstacles, the ASV is less likely to make the right decision and tends to keep the rudder unchanged. It does not respond well to the environment and prefers to crush into the border straightly. It can also be observed in Fig. 8 that the obstacle avoidance strategy has a strong correlation with the former experience. If the performance in one episode is poor, the following episode performance will show a strong continuity.

Fig. 9 shows the obstacle avoidance trajectories based on the MDQN method. By implementing the target network modification, memory pool modification, and success pool modification, the performance of present strategy is improved. Due to the limitation of rudder angle, there are difficulties in maneuvering the ASV and the optimal obstacle avoidance course is hard to be obtained. The exploration space is composed of the permutations of the ASV's velocity, sensor data, and the relative position of the ASV and the target point, which is large and complex to be traversed. Thence the value function is difficult to approximate, and the model is likely to fall into a locally optimal solution. The ASV crashes into obstacles directly on the way to the target point, as observed in Fig. 9-(c). The success rate of the ASV is relatively low. Therefore, there is volatility in the optimization process. Since the DQN method itself will lead to optimistic estimation, the algorithm also has the risk of divergence.

Fig. 10 demonstrates the obstacle avoidance trajectories based on the MDDPG method. As can be found in Fig. 10-(c), a reliable obstacle avoidance strategy can be quickly developed with around 6000 episode. Since the rudder angle is continuous, the running routes of the ASV are

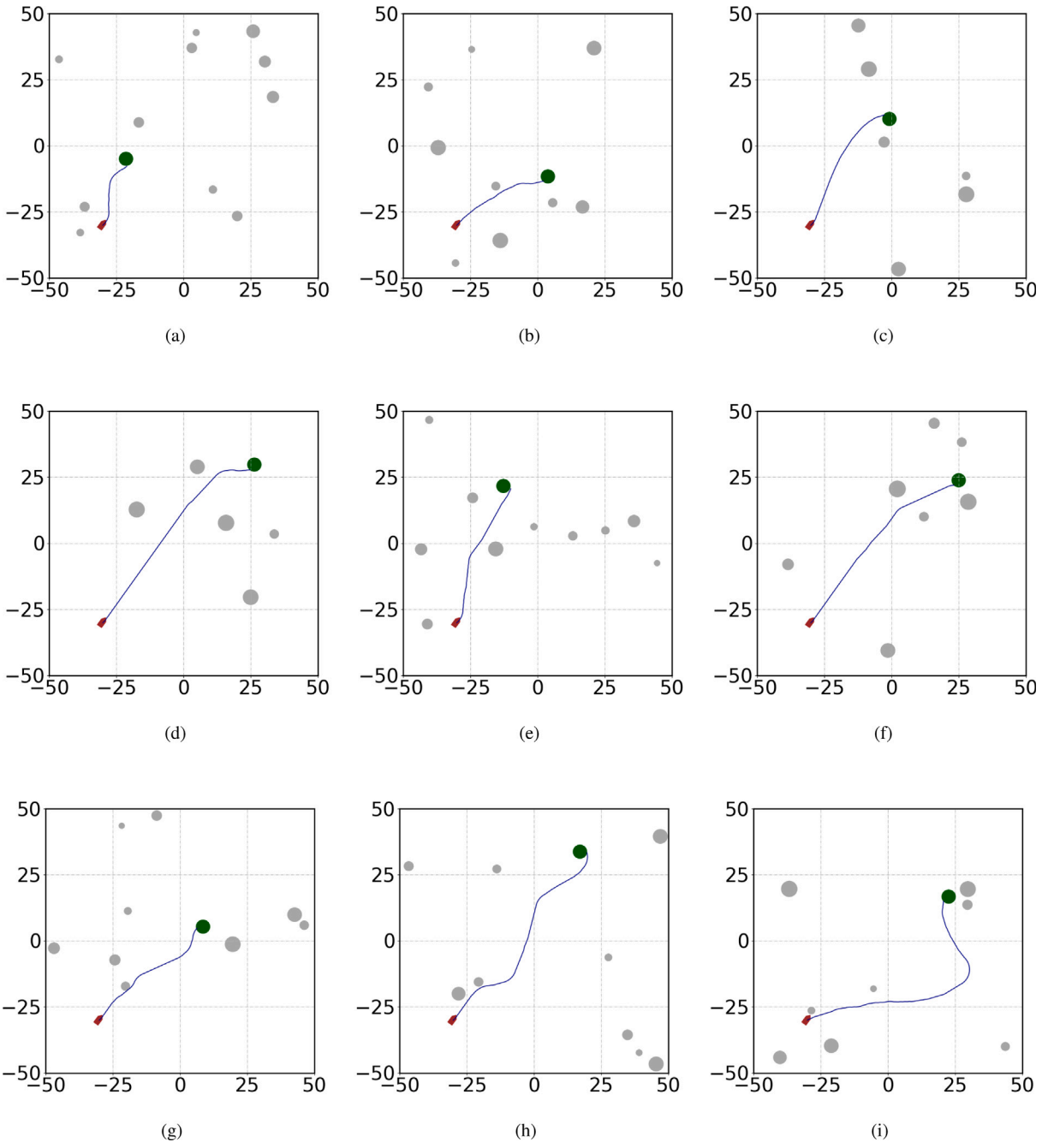


Fig. 19. Random obstacles avoidance tests using MDDPG method.

more efficient than those using DQN and MDQN. In the DQN method, limited by the selectivity of the rudder angle, the ASV often circles around the target point and cannot arrive. This problem has been well improved in MDDPG, and the ASV can approach the target point in a good posture. Because the MDDPG method obtains actions based on a strategy function, it can respond effectively when coming across an unfamiliar environmental. Therefore, the optimization process of MDDPG method is much more stable and efficient compared with the DQN method and MDQN method. The obstacle avoidance trajectory of ASV can be continuously optimized with the MDDPG method, and finally converges to a relatively constant path.

From Fig. 8-(b) to Fig. 8-(c), the trajectories of the ASV gradually turn straight and rigid. The ASV gradually loses its response to the environment. It can be concluded that without the modifications, it is easy for the RL method to diverge. For the MDQN method, though the ASV collides on the obstacle in Fig. 9-(c) for many times, the ASV still tries to avoid obstacles and drive to the target point. The strategy falls into a locally optimal solution. The limited action spaces of the MDQN method make the ASV hard to maneuver. What is more, the MDQN method is value-based, the states are acquired to be fully explored to improve the strategy. The huge state space makes it difficult for the strategy to be effectively improved.

4.2. Success rate comparison

In order to further analyze and discuss the effectiveness of the training process using the proposed methods, the success rates of the obstacle avoidance strategies using different methods are compared in this section. A successful obstacle avoidance episode is recorded when

the ASV arrives at the target position. The sliding window method is adopted to evaluate the average success rate of the latest 200 episodes.

It can be seen in Fig. 11 that the DQN method has a low success rate during the training process. In the process of training, the success rate fluctuates greatly for the DQN method. The highest success rate is about 90%, and it drops to nearly 0% for several times. It can be concluded that the method may not be suitable for ASV obstacle avoidance issue. The success rate in the MDQN method is increased quickly at the beginning as shown in Fig. 12. However, as the model falls into the optimal local solution, the success rate is decreased to around 10%, which is dangerous for the ASV navigation. As can be found in Fig. 13, the ASV is quickly trained by MDDPG method and the success rate increases to 100% with low fluctuations. In conclusion, the MDDPG method is more stable and suitable compared with other two methods.

4.3. Actions comparison

To reveal the impact of different methods on the controlling of ASV intuitively, the variations of the ASV actions with time are presented in Figs. 14 and 15.

Compared with the MDQN method, the flexibility of the action selected by the MDDPG method increases as the training progresses. The MDDPG method tends to choose the rudder angle as small as possible to complete the task. The MDDPG method reduces the variation of the rudder angle. The time consumption for DQN method decreases from nearly 300 s to 65 s, while in the MDQN method, time consumption increases, indicating that the amplitude and frequency of rudder are adjusted on a small scale. The control strategy gets more efficient than that for MDQN method. In summary, the MDDPG strategy can effectively reduce power consumption, alleviate equipment wear, and save energy.

4.4. Accumulated reward comparison

The accumulated reward R for each episode is used to represent the value of the current strategy. The higher the accumulated reward is, the higher the value of the strategy is. The reward accumulation can be obtained by multiplying the reward of each step in the whole episode by the corresponding coefficient γ. The accumulated reward function can be expressed as follows:

$$R = \sum \gamma^{i-1} r_i \tag{17}$$

The accumulated rewards of the ASV using different methods are given in Figs. 16–18. The red line represents the average accumulated reward, and the blue line denotes the accumulated reward for each episode.

Compared to the DQN method, the other two methods can effectively increase the accumulated reward. Both MDQN and MDDPG method have higher accumulated rewards than DQN method. The accumulated reward of the DQN method fluctuates with the progress of training and is less than −2300 in most of the time. In the MDQN method, as the training time increases, the fluctuation of the reward decreases, and finally converges to about −2300. The reward of the MDDPG method gradually increases and finally converges to nearly −2200. The training process of the MDDPG method is more stable and efficient than other two methods.

4.5. Random obstacle tests

To verify the applicability of the MDDPG method, random obstacles tests are also conducted. The start position and heading of the ASV are fixed in the present tests. Without loss of generality, obstacles and target positions in the environment are randomly generated in the virtual environment, and the density of obstacles is constant in this paper. The well-trained MDDPG model is implemented to the ASV. Fig. 19 demonstrate various simulation scenarios. As shown in Fig. 19-(a), (b), and (c), the ASV can arrive at the target position with a safe and efficient path. For complicated scenarios where the target position are relatively far away from the starting position with lots of obstacles nearby, as shown in Fig. 19-(d), (e), and (f), the MDDPG method remains reliable. On extreme occasions where the ASV may be blocked at the starting position as observed in Fig. 19-(g), (h), and (i), the MDDPG model helps the ASV find the way out and reaches the target.

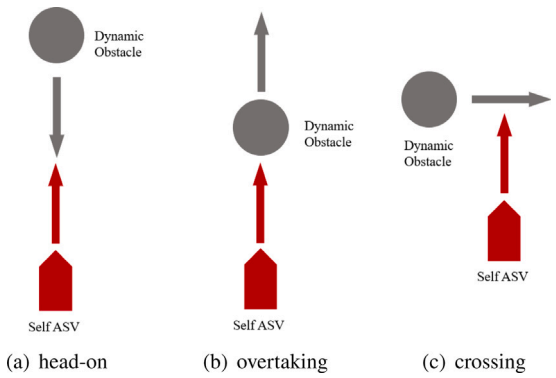


Fig. 20. Dynamic obstacles encounter situations.

Random obstacle tests further confirm that the MDDPG method has excellent applicability and generalization. In various complicated environments, ASV based on MDDPG method can effectively avoid obstacles and reach the target point.

4.6. Dynamic obstacle tests

There are three scenarios for dynamic obstacle avoidance respectively named by head-on, crossing, and overtaking. In the head-on case, the ASV encounters the obstacles which move toward it in the expected route. For crossing cases, obstacles will cross the ASV's course laterally. In overtaking cases, the obstacle and the ASV travel in the same direction, and the ASV needs to exceed the obstacle. The diagram of the scenarios is given in Fig. 20.

The training processes for dynamic obstacle avoidance are provided in Figs. 21, 22, 23 and the simulation results for dynamic obstacle tests are shown in Figs. 24, 25, 26. The dark green circle represents the target point, and the dark blue line represents the trajectory of the ASV. The position of the ASV at a certain point in time is portrait by the red outline. The yellow and green dashed lines indicate the trajectory of obstacles. The arrow shows the direction of the movement. The initial position of the obstacle is represented by the dark gray circle, and the position during the movement is represented by the light gray circles.

Head-on case:

According to COLREGS, such a situation shall be deemed to be head-on case when a vessel detects the other ahead or nearly ahead and the masthead lights of the other in a line or nearly in a line and/or both sidelights can be seen at night and by day the vessel observes the corresponding aspect of the other vessel (IMO, 1972). It is indicated from Fig. 24 that after encountering the dynamic obstacle, the ASV takes timely steering measures and effectively avoided it. After the evasion, The ASV adjusted the course and headed to the target point.

Overtaking case:

A vessel shall be deemed to be in overtaking case when coming up with another vessel from a direction more than 22.5 degrees abaft its beam, that is, in such a position with reference to the vessel is overtaking, that at night the vessel would be able to see only the stern light of that vessel but neither of its side lights (IMO, 1972). Simulation result suggests that the ASV can pass the dynamic obstacle and get to the target point successfully.

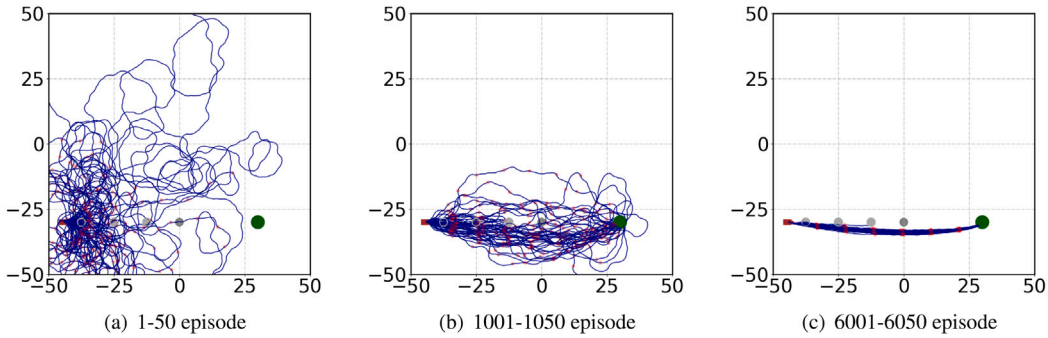


Fig. 21. The training process of head-on cases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

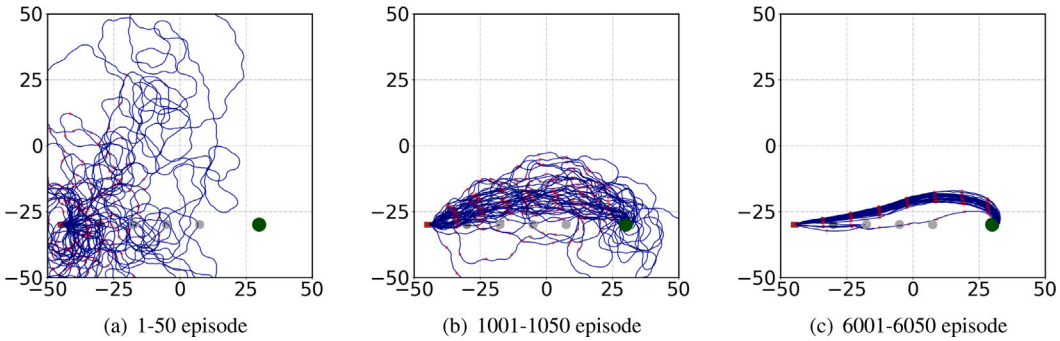


Fig. 22. The training process of overtaking cases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

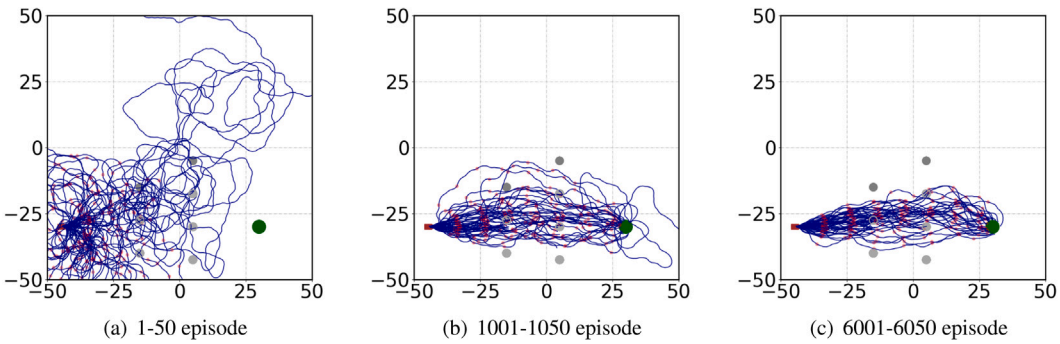


Fig. 23. The training process of crossing cases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

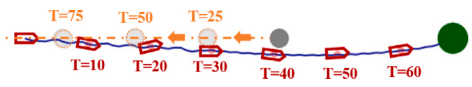


Fig. 24. Dynamic obstacle avoidance test for the head-on scenario. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

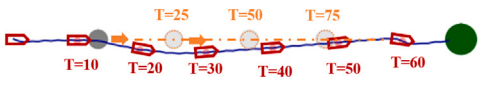


Fig. 25. Dynamic obstacle avoidance test for the overtaking scenario. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

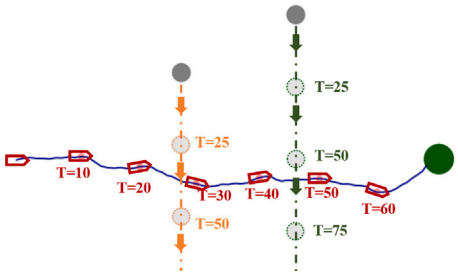


Fig. 26. Dynamic obstacle avoidance test for the crossing scenario. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Crossing case:

In the crossing case, two obstacles are set up to cross the expected course of the ASV. The ASV made a timely avoidance after sensing the obstacle, and finally reached the target point. As shown in Fig. 26, the ASV adjusted its direction for several times. Since the input of the algorithm is instant, the perception range of ASV does not include the environmental information of the previous time period. Therefore the ASV is completely ignorant of the speed and direction of dynamic obstacles, all obstacle avoidance strategies are implemented based on the current situation. The ASV adopts a strategy that is inclined to drive

toward the target point when it considers the environment safe and is inclined to ensure its own safety when it is at risk. The reason why the ASV varies several times of direction in the crossing case is that, after adjusting the course, the ASV is considered to be in a safe area and start driving straight to the target point. Then after a while, ASV may encounter the obstacles that it had evaded before, and readopt a more conservative strategy.

Algorithm 1 Deep Q Network

Require: network learning rate lr, random probability ϵ, random probability decrease factor c, hyperparameter γ
Initialize actor-network with random weight θ
for each round of simulate **do**
for each step i **do**
Observe the environment and obtain the state s. Generate action a_o by $a_o = argmax(Q_\theta(s, a_o))$
random state $ra = random(0, 1)$
if $ra < \epsilon$ **then**
Randomly choose an action a from the three action spaces.
else
$a = a_o$
end if
$\epsilon = \epsilon \times c$
take action a and get the next state s' and reward r.
Update Q-funct=ion by gradient descent using
$\triangle = \bigtriangledown_\theta (Q_\theta(s, a) - max(r + \gamma Q_{\theta^-}(s', a'))^2$
$\theta \leftarrow \theta + lr \cdot \triangle$
end for
end for

The simulation results imply that the proposed RL strategy can still guarantee the safety and reliability of the ASV when it comes to dynamic obstacles based on real-time information. However, because of lacking speed and direction information of dynamic obstacles, The MDDPG method may not efficiently be suitable for dynamic obstacle avoidance problems. During the voyage, there are multiple turns to ensure safety and reliability rather than just a few turns. However, in real scenarios, dynamic obstacles usually do not move in a straight line at a uniform speed, nor do they communicate with ASV. The MDDPG method based on instant information may show a better adaptability.

5. Conclusion

An obstacle avoidance strategy based on MDDPG method is proposed in this paper. Target network modification, memory pool modification, and success pool modification are developed to improve the capability of the neural network. Compared with the DQN and MDQN method, training process of the MDDPG method is smoothed and accelerated. The continuous rudder angles can be chosen due to the policy gradient method. The obstacle avoidance in a complicated environment can be realized by the MDDPG method.

The simulation results indicate that the MDDPG method makes a smoother optimization process and a more stable action-generate model than other two methods. The capability of the MDDPG method has also been validated when it comes to dynamic obstacle issues. Benefit from the continuous action-generate function, better trajectory planning capabilities can be obtained. Compared with the DQN method and the MDQN method, a higher success rate can be achieved by the MDDPG method, meaning a higher reliability, flexibility and safety in practical applications. The amplitude and frequency of rudder adjustment are decreased, reducing power consumption and the equipment wear. It is shown from random obstacle tests that the MDDPG method has excellent applicability and generalization under various complex environments.

In future research, more accurate ship dynamics models and marine environment interference models will be introduced. The effects of wind, wave, and current will be taken into account in simulations.

Algorithm 2 Modified Deep Q Network

Require: network learning rate lr, minibatch size k, memory pool size N_{memory}, success pool size $N_{success}$, empty memory pool $\mathcal{M}$, empty success pool $\mathcal{S}$, random probability ϵ, random probability decrease factor c, hyperparameter ρ, γ
Initialize actor-network with random weight θ
Initialize target actor-network with weight $\theta^- = \theta$
for each round of simulate **do**
for each step i **do**
Observe the environment and obtain the state s. Generate action a_o by $a_o = argmax(Q_\theta(s, a_o))$
random state $ra = random(0, 1)$
if $ra < \epsilon$ **then**
Randomly choose an action a from the three action spaces.
else
$a = a_o$
end if
$\epsilon = \epsilon \times c$
take action a and get the next state s' and reward r, store transition (s, a, s', r) in memory pool, and store the transition in the temporary list T.
if memory pool $\mathcal{M}$ is full **then**
Replace the oldest data with the new transition.
end if
sample minibatch of transitions $\mathcal{B}$ from memory pool $\mathcal{M}$ and success pool $\mathcal{S}$.
Update Q-function by gradient descent using
$\triangle = \bigtriangledown_\theta \frac{1}{|\mathcal{B}|} \sum_{(s,a,s',r)\in\mathcal{B}} (Q_\theta(s, a) - max(r + \gamma Q_{\theta^-}(s', a'))^2$
$\theta \leftarrow \theta + lr \cdot \triangle$
Update the target network
$\theta^- \leftarrow \rho\theta^- + (1 - \rho)\theta$
end for
if The ASV arrive the target point **then**
Store list T to success pool $\mathcal{S}$
if success pool $\mathcal{S}$ is full **then**
Replace the oldest data with the new transition.
end if
else
Empty list T
end if
end for

The obstacle avoidance capability of the proposed strategy will be further verified and validated by indoor model experiments and field tests. A neural network with memory capabilities to strengthen the ASV's dynamic obstacle avoidance ability will be introduced in the future. Besides, the intelligent obstacle avoidance strategy compliant with COLREGS will be considered in our future work.

CRediT authorship contribution statement

Chang Zhou: Conceptualization, Methodology, Software, Data curation, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Yiting Wang:** Methodology, Investigation, Supervision, Project administration, Funding acquisition. **Lei Wang:** Conceptualization, Methodology, Software, Investigation, Writing – Review & Editing. **Huacheng He:** Methodology, Software, Investigation, Writing – review & editing, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors greatly acknowledge the support of the National Natural Science Foundation of China (Grant No. 51179103), the Ministry of Industry and Information Technology (Grant No. [2018] 473) and the Hainan Provincial Joint Project of Sanya Bay Science and Technology City (Grant No. 520LH051).

Appendix. Algorithm adopted in the present paper

Algorithm 3 Modified Deep Deterministic Policy Gradient

Require: actor-network learning rate l_a, critic-network learning rate l_c, minibatch size k, memory pool size N_{memory}, success pool size $N_{success}$, empty memory pool $\mathcal{M}$, empty success pool $\mathcal{S}$, random deviation σ, intercept d, intercept decrease factor c, hyperparameter ρ, γ

Initialize actor-network with random weight θ_a

Initialize target actor-network with weight $\theta_a^- = \theta_a$

Initialize critic-network with random weight θ_c

Initialize target critic-network with weight $\theta_c^- = \theta_c$

for each round of simulate **do**

  for each step i **do**

    Observe the environment and obtain the state s. Generate action a_o by $a_o = \mu_{\theta_a}(s)$

    $a_{random} = random(normal(a_o, \sigma))$

    if $a_{random} < a_o - d$ **then**

      $a = a_o - d$

    else if $a_{random} > a_o + d$ **then**

      $a = a_o + d$

    else

      $a = a_{random}$

    end if

    $d = d \times c$

    take action a and get the next state s' and reward r, store transition (s, a, s', r) in memory pool, and store the transition in the temporary list T.

    if memory pool $\mathcal{M}$ is full **then**

      Replace the oldest data with the new transition.

    end if

    sample minibatch of transitions $\mathcal{B}$ from memory pool $\mathcal{M}$ and success pool $\mathcal{S}$.

    Update Q-function by gradient descent using

    $\triangle_c = \nabla_{\theta_c} \frac{1}{|\mathcal{B}|} \sum_{(s,a,s',r)\in\mathcal{B}} (Q_{\theta_c}(s,a) - (r + \gamma Q_{\theta_c^-}(s', a')))^2$

    $\theta_c \leftarrow \theta_c + lc \cdot \triangle_c$

    Update policy by gradient ascent using

    $\triangle_a = \nabla_{\theta_a} \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}} Q_{\theta_c}(s, \mu_{\theta_a}(s))$

    $\theta_a \leftarrow \theta_a + la \cdot \triangle_a$

    Update the target actor-network and critic-network

    $\theta_c^- \leftarrow \rho\theta_c^- + (1-\rho)\theta_c$

    $\theta_a^- \leftarrow \rho\theta_a^- + (1-\rho)\theta_a$

  end for

  if The ASV arrive the target point **then**

    Store list T to success pool $\mathcal{S}$

    if success pool $\mathcal{S}$ is full **then**

      Replace the oldest data with the new transition.

    end if

  else

    Empty list T

  end if

end for

References

Bellman, R., 1957. A Markovian decision process. J. Math. Mech. 6 (5), 679–684.

Chen, C., Chen, X.-Q., Ma, F., Zeng, X.-J., Wang, J., 2019. A knowledge-free path planning approach for smart ships based on reinforcement learning. Ocean Eng. 189, 106299.

Cheng, Y., Zhang, W., 2018. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing 272, 63–73.

Dijkstra, E.W., et al., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

Doran, J.E., Michie, D., 1966. Experiments with the graph traverser program. Proc. R. Soc. Lond. Ser. A 294 (1437), 235–259.

Fossen, T.I., 2002. Marine Control Systems–Guidance. Navigation, and Control of Ships, Rigs and Underwater Vehicles. Marine Cybernetics, Trondheim, Norway, ISBN: 82 92356 00 2, Org. Number NO 985 195 005 MVA, www.marinecybernetics.com.

Fox, D., Burgard, W., Thrun, S., 1997. The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. 4 (1), 23–33.

Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J.M., 1989. Self-organized shortcuts in the argentine ant. Naturwissenschaften 76 (12), 579–581.

IMO, 1972. Convention on the International Regulations for Preventing Collisions at Sea. IMO London.

Łącki, M., 2008. Reinforcement learning in ship handling. TransNav: Int. J. Mar. Navig Safety Sea Transp. 2 (2).

Layek, A., Vien, N., Chung, T., et al., 2017. Deep reinforcement learning algorithms for steering an underactuated ship. In: 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). IEEE, pp. 602–607.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lozano-Perez, T., Mason, M.T., Taylor, R.H., 1984. Automatic synthesis of fine-motion strategies for robots. Int. J. Robot. Res. 3 (1), 3–24.

Lozano-Pérez, T., Wesley, M.A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM 22 (10), 560–570.

Martinsen, A.B., Lekkas, A.M., 2018. Straight-path following for underactuated marine vessels using deep reinforcement learning. IFAC-PapersOnLine 51 (29), 329–334.

Masoud, S.A., Masoud, A.A., 2000. Constrained motion control using vector potential fields. IEEE Trans. Syst. Man Cybern. A 30 (3), 251–272.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Shen, H., Hashimoto, H., Matsuda, A., Taniguchi, Y., Terada, D., Guo, C., 2019. Automatic collision avoidance of multiple ships based on deep Q-learning. Appl. Ocean Res. 86, 268–288.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms. In: International Conference on Machine Learning. PMLR, pp. 387–395.

Skjetne, R., Smogeli, Ø., Fossen, T.I., 2004. Modeling, identification, and adaptive maneuvering of Cybership II: A complete design with experiments. IFAC Proc. Vol. 37 (10), 203–208.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Voronoi, G., 1908. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. J. Reine Angew. Math. 1908 (134), 198–287.

Wang, C., Zhang, X., Cong, L., Li, J., Zhang, J., 2018. Research on intelligent collision avoidance decision-making of unmanned ship in unknown environments. Evol. Syst.

Woo, J., Kim, N., 2020. Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. Ocean Eng. 199, 107001.

Woo, J., Yu, C., Kim, N., 2019. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Eng. 183, 155–166.

Xu, X., Lu, Y., Liu, X., Zhang, W., 2020. Intelligent collision avoidance algorithms for USVs via deep reinforcement learning under COLREGs. Ocean Eng. 217, 107704.

Yoo, B., Kim, J., 2016. Path optimization for marine vehicles in ocean currents using reinforcement learning. J. Mar. Sci. Technol. 21 (2), 334–343.

Zhang, X., Wang, C., Jiang, L., An, L., Yang, R., 2021. Collision-avoidance navigation systems for Maritime Autonomous Surface Ships: A state of the art survey. Ocean Eng. 235, 109380.

Zhao, L., Roh, M.-I., 2019. COLREGs-compliant multiship collision avoidance based on deep reinforcement learning. Ocean Eng. 191, 106436.

Zhao, L., Roh, M.-I., Lee, S.-J., 2019. Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. J. Mar. Sci. Technol. 27 (4), 1.