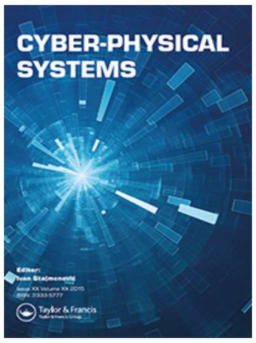


Cyber-Physical Systems

ISSN: 2333-5777 (Print) 2333-5785 (Online) Journal homepage: http://www.tandfonline.com/loi/tcyb20

Deep reinforcement learning-based path planning of underactuated surface vessels

Hongwei Xu, Ning Wang, Hong Zhao & Zhongjiu Zheng

To cite this article: Hongwei Xu, Ning Wang, Hong Zhao & Zhongjiu Zheng (2018): Deep reinforcement learning-based path planning of underactuated surface vessels, Cyber-Physical Systems, DOI: 10.1080/23335777.2018.1540018

To link to this article: https://doi.org/10.1080/23335777.2018.1540018

Published online: 14 Nov 2018.

CYBER-PHYSICAL SYSTEMS
https://doi.org/10.1080/23335777.2018.1540018

ARTICLE

Deep reinforcement learning-based path planning of underactuated surface vessels

Hongwei Xu[a], Ning Wang[b], Hong Zhao[b] and Zhongjiu Zheng[b]

[a]Marine Engineering College, Dalian Maritime University, Dalian, China; [b]School of Marine Electrical Engineering, Dalian Maritime University, Dalian, China

ABSTRACT
This paper is dedicated to the issues of path planning for the underactuated surface vessel (USV) in unknown environments with obstacles. Aiming at the usability problem caused by the complicated control law of the traditional method, a deep deterministic policy gradient (DDPG)-based path planning algorithm is proposed with the powerful actor-critic architecture. The main contributions of this paper are as follows: (1) The DDPG-based path planning method is the decision control system of USV whose output is continuous. (2) A complete reward system is specially designed for target approaching, speed control and attitude correction. (3) Since the reward system is independent, it is highly customisable and can be changed according to the task and the control model. Through the simulation, the results show that the perfect path can be automatically generated under unknown environmental disturbance, thus providing the proposed DDPG-based path planning scheme effectiveness and applied meaning.

ARTICLE HISTORY
Received 18 October 2018
Accepted 19 October 2018

KEYWORDS
Path planning; deep deterministic policy gradient; deep reinforcement learning; underactuated surface vessel

1. Introduction

With increasing concern of the exploitation of marine resources and maritime rights, people have higher expectations and requirements on the realisation of intelligent control and autonomous navigation of ships. The underactuated vessel has the advantages of simple structure design of actuator, low operation cost, less mechanical failure rate and strong system reliability. Therefore, the thorough study of underactuated vessel motion control not only promotes the development of vessels intelligent and automatic but also provides technical support for ship transportation, marine resource development and utilisation and national defence military construction [1]. In addition, the navigation track planning and intelligent obstacle avoidance technology are directly related to the automation and intelligence of navigation control [2]. In 1959, researcher began to study path planning to maximise economic benefits [3]. After that, a

CONTACT Ning Wang n.wang.dmu.cn@gmail.com School of Marine Electrical Engineering, Dalian Maritime University, Dalian, China

large number of researchers have done lots of positive research on path planning and obstacle avoidance and proposed many planning algorithms [4–12]. However, there are some problems to be solved: (1) The ability to deal with complex environments is weak or even cannot be dealt with. (2) It can only be designed independently according to specific circumstances and lacks of portability.

On the other hand, with the rapid development of artificial intelligence (AI), researchers in various fields are breaking through the technical bottleneck continuously. In [13,14], using deep reinforcement learning (DRL) lets computers learn to play classic Atari games and in some games scores higher than humans. David Silver and Aja Huang [15] designed AlphaGo to beat the human go champion using DRL [16]. From this, it can be seen that DRL has a strong ability to receive and process complex environmental information. Therefore, in [17], using DRL as a decision-making system, underactuated surface vessel (USV) receives complex information from the environment. After several self-learning, it can plan a reasonable route and safely avoid all obstacles to reach its destination. DRL is also applicable in three-dimensional environment. Unmanned aerial vehicle can plan the route to the terminal while avoiding collision [18]. Nevertheless, there are still problems: the output of the decision-making system is discrete, which may lead to collision with obstacles in the process of agent motion because the discrete action cannot be avoided in time.

From the perspective of system design, the guidance and control problems in USV motion control can include path planning, path following and autonomous obstacle avoidance. Under the guidance and control, the off-line global route planning is carried out first. Then, real-time avoidance of unknown obstacles can be realised in the following process, while following the planned path [19,20]. Finally, the USV autonomous navigation and operation mission under the unknown complex marine environment were completed. In this paper, the hidden combination of the three is used for path planning and real-time path following and autonomous obstacle avoidance through the vessel dynamics model and various sensors on the vessel. In this paper, the three elements are implicit integrated together. In the process of path planning, real-time path following and obstacle avoidance are carried out through vessel kinetic model and various sensors to explore the surrounding environment.

The main content of this paper is that USV which is unknown to the environment can reach the destination without collision from the initial position by DRL technique. The output of the decision module of the proposed deep deterministic policy gradient (DDPG)-based path planning algorithm is continuous action, which makes the path planned more optimised and reasonable. Further, it easy to transplant to other models because the decision module and the USV model are relatively independent modules, greatly increasing the portability and ease of use of the system.

The rest of the paper is as follows: Section 2 illustrates the system description. Each part of the algorithm and the reward function are addressed in Sections 3 and 4, respectively. Simulation studies are conducted in Section 5. Section 6 concludes this paper.

2. System description

2.1. Preliminaries

In this note, the study of path planning takes USVs as the example. The horizontal position state vector of USV is defined as $\boldsymbol{\eta} = [x \quad y \quad \psi]^T$ where (x, y) is the position in the earth-fixed frame and ψ is the azimuth. $v = [u \quad v \quad r]^T$ are chosen as the velocity vector and u, v and r represent surge velocity, sway velocity and yaw rate, respectively. Figure 1 illustrates the various quantity of the vessel in motion, where r_o is vessel detection radius.

In order to design the whole path planning system, the ship model with six degrees of freedom needs to be simplified as follows:

(1) The surge, roll and yaw are considered, while the heave, roll and pitch are ignored.
(2) The vessel is symmetric about OXZ plane ($I_{xy} = I_{yz} = 0$) and has uniform mass distribution.

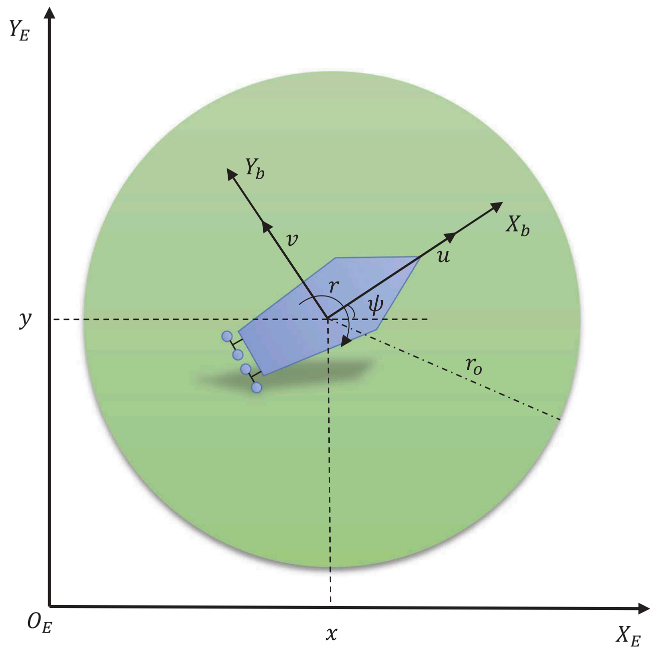


Figure 1. Planar motion model of an underactuated surface vessel.

(3) The origin of the hull coordinate system is the central point of the vessel, that is, the centre of the vessel and the centre of gravity coincide.
(4) In general, USVs mean that the ship cannot produce thrust laterally (or the transverse thruster cannot work properly due to other reasons), only has thruster longitudinally, so the thrust vector $\boldsymbol{\tau}$ expression is $\boldsymbol{\tau} = \left[\boldsymbol{\tau}_u \quad 0 \quad \boldsymbol{\tau}_v\right]^T$. $\boldsymbol{\tau}_u$ and $\boldsymbol{\tau}_t$ are the trust moment and turning moment, respectively. Since the USV has no lateral thruster force, the value is zero.
(5) Ignoring the hydrodynamic terms higher than the first order.

Based on the above conditions, the three-degree-of-freedom model of the horizontal plane of the non-linear underactuated vessel can be obtained as follows [21]:

$$\begin{cases} \dot{\boldsymbol{\eta}} & = J(\boldsymbol{\eta})v \\ \boldsymbol{\tau} + \boldsymbol{\tau}_w & = \mathbf{M}\dot{v} + \mathbf{C}(v)v + \mathbf{D}(v)v \end{cases} \tag{1}$$

where

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

and $\boldsymbol{M}$ is the inertia matrix (including the added mass) and positive definite matrix, $M = M^T$

$$\begin{aligned} \boldsymbol{M} &= \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix} \\ &= \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & I_z - N_{\dot{r}} \end{bmatrix} \end{aligned} \tag{3}$$

and $C(v)$ is the Coriolis and centripetal matrix, and it has the properties that $C(v) = -C^T(v)$ as skew symmetric matrix.

$$\boldsymbol{C}(\boldsymbol{v}) = \begin{bmatrix} 0 & 0 & C_{13} \\ 0 & 0 & C_{23} \\ -C_{13} & -C_{23} & 0 \end{bmatrix} \tag{4}$$

with

$$C_{13} = -m_{22}v - m_{23}v \tag{5}$$

$$C_{23} = m_{11}u \tag{6}$$

and $\boldsymbol{D}(\boldsymbol{v})$ is the damping matrix.

$$\begin{aligned}\boldsymbol{D}(\boldsymbol{v}) &= \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \\ &= \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & 0 \\ 0 & 0 & -N_r \end{bmatrix}\end{aligned} \tag{7}$$

where X, Y and N are the hydrodynamic derivatives.

The time-varying disturbance from the environment can be expressed as [19]:

$$\boldsymbol{\tau}_w = 3\begin{bmatrix} \sin t \\ \cos t \\ \cos t \end{bmatrix} - 0.2\begin{bmatrix} vr \\ ur \\ uv \end{bmatrix} \tag{8}$$

2.2. Problem function

In the whole path planning system, our main task is to calculate an optimal route to reach the end point and avoid collision in the unknown environment. The final planned path is actually USV control sequence output by a decision system because USV is used as the motion model in this paper. The decision-making system whose details can be found in Section 3 is the core of DRL. Therefore, its input X_t is not only the current state s_t but also the state in the historical period s_{t-i}, $i = 1, \cdots, T_b$, where T_b represents the length of the historical state sequence that the decision-making system needs to consider. Through the full study of navigation, the definition of state quantity is as follows:

$$\boldsymbol{\tau}_t = f_{DMS}(\boldsymbol{X}_t) = f_{DMS}(s_t, s_{t-1}, \cdots, s_{t-T_b}) \tag{9}$$

$$s_t = \begin{bmatrix} \sigma_t & \boldsymbol{\eta}_t & \boldsymbol{v}_t & \boldsymbol{\tau}_{t-1} \end{bmatrix} \tag{10}$$

where f_{DMS} is intelligent decision system function and σ_t is a binary number, which represents whether there is any obstacle within the vessel detection area.

3. DDPG algorithm

By processing the motion state of the vessel and current environment information through the decision-making system, a series of continuous control sequences of the vessel can be obtained. The trajectory is the planned path. Therefore, it is essential for the collection of the current state of the vessel, including the position of the vessel (x, y), the heading ψ and the velocity quantity u, v and r. In addition, the vessel detection system detected obstacles

σ_t within a certain range at time t and the last control behaviours $\boldsymbol{\tau}_{t-1}$. Consequently, the observation $\boldsymbol{s}_t = \left[\sigma_t \quad \boldsymbol{\eta}_t \quad \boldsymbol{v}_t \quad \boldsymbol{\tau}_{t-1}\right]$ can present the information of vessel and environment at time t comprehensively. The input of the decision-making system also includes historical information $\boldsymbol{s}_{t-i}, i = 1, \cdots, T_b$, but the direct input of a large amount of data will lead to DRL's poor learning. So, data compression module is very necessary which integrates current and historical information and extracts features for the decision system to learn. The historical information and current information provided by the decision system through the data compression module use DRL approach to learn the control behaviour sequence.

This section mainly describes the details of each part of the proposed DDPG-based path planning algorithm and the connections between the components.

3.1. Data compression module

Convolutional neural network (CNN) is an efficient identification method which has been developed in recent years. In 1962, Hubel and Wiesel [22] studied the cat's visual cortex systematically and found that each neuron could only process a small area of visual images and then proposed the concept of receptive field. Fukushima [23] proposed Neocognitron based on the concept of receptive field in the mid-1980s. Since then, researchers have tried to use multilayer perceptron instead of manual feature extraction. Until 1990, Lecun [24] first proposed a true convolution neural network.

Weight sharing and sparse connectivity are two major features of CNNs. Sparse connection refers to digging the spatial local correlation information of natural images by strengthening the local connection mode of nodes between adjacent layers in the neural network. Each neuron does not respond to its sensory field changes, and this structure ensures that each trained filter only responds most strongly to local spatial input patterns. The multilayered neuron accumulation forms a non-linear filter, which is progressively more global (response to larger input regions). Each convolution filter are repeated throughout the visual area, namely the filter is convolved with the feature map of the previous layer, the parameters of the filter is the same, and thus for the result of the convolution is sharing the parameters (weights and bias) [24]. Repetitive filters can detect features without considering location information, and sharing weights can greatly reduce the number of free parameters to be learned, thus improving learning efficiency.

If we denote x the as the input data and f^k as the kth feature map, while convolutional filter and bias used W^k and b_k, respectively. On the other hand, since the activation function should be a non-linear function in this paper, we can use Rectified Linear Units (ReLU) which is proved to be more supportive [25,26]. So, the calculation process can be expressed as

$$f_{ij}^{k} = Activation\left(\left(W^{k} \otimes x\right)_{ij} + b_{k}\right) \tag{11}$$

where $\otimes$ denotes the convolutional operator.

$$\begin{aligned} o[m,n] &= h[m,n] \otimes g[m,n] \\ &= \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} h[u,v] g[m-u, n-v] \end{aligned} \tag{12}$$

The use of multiple single observation vectors can make data encoding more effective. In our system, the input data include only the current s_t but also T_b historical observation vectors $s_{t-i}, i \in 1, \cdots, T_b$. Consequently, the input for the decision-making system at the moment t can be expressed as

$$\begin{aligned} \boldsymbol{X}_t &= [\boldsymbol{s}_t \quad \boldsymbol{s}_{t-1} \quad \ldots \quad \boldsymbol{s}_{t-T_b}]^T \\ &= \begin{bmatrix} \sigma_t & \boldsymbol{\eta}_t & \boldsymbol{v}_t & \boldsymbol{\tau}_{t-1} \\ \sigma_{t-1} & \boldsymbol{\eta}_{t-1} & \boldsymbol{v}_{t-1} & \boldsymbol{\tau}_{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{t-T_b} & \boldsymbol{\eta}_{t-T_b} & \boldsymbol{v}_{t-T_b} & \boldsymbol{\tau}_{t-T_b-1} \end{bmatrix} \end{aligned} \tag{13}$$

After convolution layer by layer, each input observation matrix is mapped into a complete state vector which is used to describe the current state of the USV. In the meantime, the historical information on control behaviours and the existence of obstacles are taken into account.

3.2. Decision-making system

The decision system uses the features extracted from the data compression module as the output and continuously uses DRL method to explore the implicit optimisation control behaviour in the unknown environment. In fact, DRL can be divided into two parts: deep learning (DL) and reinforcement learning (RL). The DL is not good at solving automatic control problems because it focuses on the function approximation of high-dimensional non-linearity and the acquisition of high-order abstract features. The RL is an optimisation process of control strategy. The combination of DL and RL, namely DRL, is the application of DL in the aspect of RL. For some relatively complex control problems, the application of RL algorithm of deep neural network can achieve good results. In this paper, the intelligent agent, state and action in DRL are the USV, X_t and $\boldsymbol{\tau}_t$, respectively.

When we study RL and control problems, an agent selects the control behaviour sequentially under a series of time in the unknown environment, in order to maximise a cumulative reward. Therefore, this is so-called the Markov decision process (MDP): a state space $\boldsymbol{S}$, an action space $\boldsymbol{A}$, a reward function R and transition function P_{sa}. At each time t, the agent observes the current state $\boldsymbol{X}_t \in \boldsymbol{S}$. Then, an action $\boldsymbol{\tau}_t \in \boldsymbol{A}$ is selected to generate a transition from current state $\boldsymbol{X}_t$ to new state $\boldsymbol{X}_{t+1}$. The agent gets an immediate reward $r_t = R(\boldsymbol{X}_t, \boldsymbol{\tau}_t)$ in the process. In addition, for any trajectory $\boldsymbol{X}_1, \boldsymbol{\tau}_1, \boldsymbol{X}_2, \boldsymbol{\tau}_2, \cdots, \boldsymbol{X}_t, \boldsymbol{\tau}_t$ in state–action space, the transition P_{sa} satisfies the Markovian property:

$$p_{sa}(\boldsymbol{X}_{t+1}|\boldsymbol{X}_1, \boldsymbol{\tau}_1, \cdots, \boldsymbol{X}_t, \boldsymbol{\tau}_t) = p_{sa}(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t, \boldsymbol{\tau}_t) \tag{14}$$

In MDP, the total discounted reward from time step t onwards is defined as:

$$\begin{aligned} R(h_t) &= \gamma^0 R(\boldsymbol{X}_t, \boldsymbol{\tau}_t) + \gamma^1 R(\boldsymbol{X}_{t+1}, \boldsymbol{\tau}_{t+1}) + \cdots \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} R(\boldsymbol{X}_k, \boldsymbol{\tau}_k) \end{aligned} \tag{15}$$

where $\gamma \in (0, 1)$ is discount factor.

A policy is used to select actions in MDP. In general, the policy is stochastic and can be expressed as $\pi : \boldsymbol{S} \longrightarrow P(\boldsymbol{A})$, where $P(\boldsymbol{A})$ is the set of probability measures on $\boldsymbol{A}$ and $\pi(\boldsymbol{\tau}_t|\boldsymbol{X}_t)$ is the conditional probability density at $\boldsymbol{\tau}_t$ associated with the policy. The goal of the agent is to obtain a policy that maximises the total discount reward from the start state, which denoted by the performance objective [27]:

$$J(\pi) = \mathbb{E}[R(h_1)|\pi] \tag{16}$$

In DDPG-based path planning algorithm, we use the state–action value function to determine whether the control strategy is good or bad, and it is also the expectation of total discount rewards.

$$Q^{\pi}(\boldsymbol{X}, \boldsymbol{\tau}) = \mathbb{E}[R(h_1)|\boldsymbol{X}_t = \boldsymbol{X}, \boldsymbol{\tau}_1 = \boldsymbol{\tau}; \pi] \tag{17}$$

Combining Equations (15) and (17), we can obtain the following Bellman equation:

$$\begin{aligned} Q^{\pi}(\boldsymbol{X}_t, \boldsymbol{\tau}_t) = \quad & R(X_t, \boldsymbol{\tau}_t) + \\ & \mathbb{E}_{\pi(\boldsymbol{\tau}_{t+1}|X_{t+1}) p_{sa}(X_{t+1}|X_t, \boldsymbol{\tau}_t)}[Q^{\pi}(\boldsymbol{X}_{t+1}, \boldsymbol{\tau}_{t+1})] \end{aligned} \tag{18}$$

Suppose $\rho^{\pi}(\boldsymbol{X}_i) = P_{sa}(\boldsymbol{X}_{i+1}|\boldsymbol{X}_i, \boldsymbol{\tau}_i)$, Equation (16) can be rewritten:

$$\begin{aligned} J(\pi) &= \int_S \rho^{\pi}(\boldsymbol{X}) \int_A \pi(\boldsymbol{\tau}|\boldsymbol{X}) Q^{\pi}(\boldsymbol{X}, \boldsymbol{\tau}) d\boldsymbol{\tau} d\boldsymbol{X} \\ &= \mathbb{E}_{\boldsymbol{X} \sim \rho^{\pi}, \boldsymbol{\tau} \sim \pi}[Q^{\pi}(\boldsymbol{X}, \boldsymbol{\tau})] \end{aligned} \tag{19}$$

Once again, the control behaviours of agent using the algorithm are continuous rather than discrete, which makes agent more flexible and less dangerous. Therefore, DDPG method [28] is adopted as the decision-making system that is the core of DRL. DDPG is an improvement of deterministic policy gradient. It imports the experience replay and dual network mechanisms into the actor-critic algorithm framework to obtain the strategy learning method for continuous behaviour. Figure 2 illustrates the framework and operational process of the DDPG method.

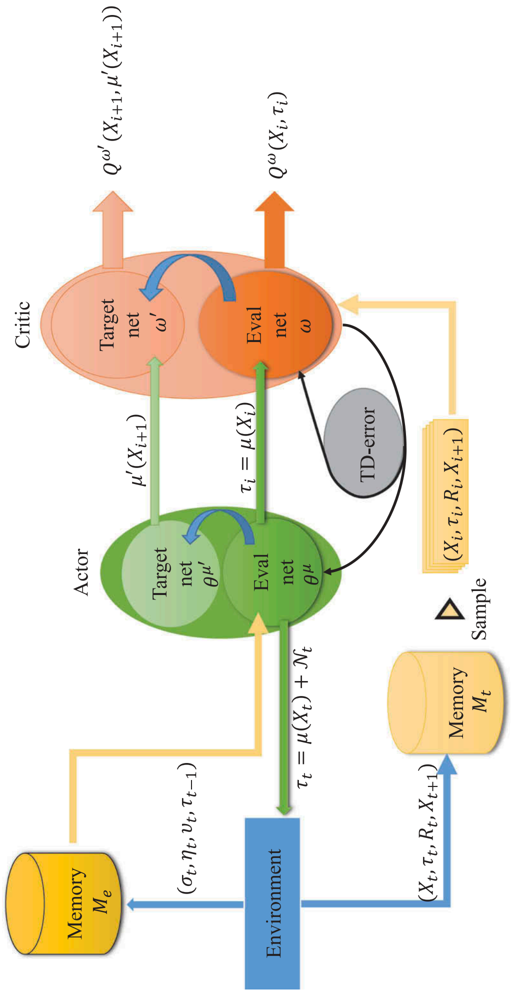

Critic
Target net ω'
Eval net ω
$Q^{\omega'}(X_{i+1}, \mu'(X_{i+1}))$
$Q^{\omega}(X_i, \tau_i)$
Actor
Target net $\theta^{\mu'}$
Eval net θ^{μ}
$\mu'(X_{i+1})$
$\tau_i = \mu(X_i)$
TD-error
$(X_i, \tau_i, R_i, X_{i+1})$
Sample
Memory M_t
Memory M_e
Environment
$\tau_t = \mu(X_t) + \mathcal{N}_t$
$(\sigma_t, \eta_t, \upsilon_t, \tau_{t-1})$
$(X_t, \tau_t, R_t, X_{t+1})$

Figure 2. DDPG framework.

Experience replay refers to the establishment of an experience pool. The system store information such as state, action and reward in the memory pool continuously when the agent is interacting with the environment. In our algorithm, there are two different experience pools, M_t and M_e, which are used to store T_b historical observation $\boldsymbol{s}_{t-i} = [\sigma_{t-i} \quad \boldsymbol{\eta}_{t-i} \quad \boldsymbol{v}_{t-i} \quad \boldsymbol{\tau}_{t-1-i}]$, $i = 0, 1, \cdots, T_b$ and N_t transition quadruples $(\boldsymbol{X}_t, \boldsymbol{\tau}_t, R_t, \boldsymbol{X}_{t+1})$, respectively.

Dual network mechanism refers to the network that constructs two same structures with different parameters, namely target network and evaluation network. The evaluation network is actually involved in operation, and the target network does not participate in the interaction with the environment but is only used during training. The evaluation network will pass its parameters to the target network in the following ways within a certain interval [28].

$$\begin{aligned} \theta^{\mu'} &\leftarrow \varepsilon\theta^{\mu} + (1-\varepsilon)\theta^{\mu'} \\ \omega' &\leftarrow \varepsilon\omega + (1-\varepsilon)\omega' \end{aligned} \tag{20}$$

where $\theta^{\mu'}$ and θ^{μ} are the target and evaluation parameters in actor. ω' and ω are critic parameters. ε is conversion rate.

Combined with the above information and Figure 2, actor and critic have two networks with the same initial parameters. The actor-evaluation network receives historical observations from the experience pool M_e and uses the data compression module to output control behaviours through the decision system, which is continuous rather than discrete. Furthermore, control behaviours should be coupled with noise N_t [29,30], exploding and exploiting the latent optimal control behaviours. There is a penalty equation in the environment, which is designed on the basis of tasks. Although it is not certain, it must satisfy a condition: it can implicitly contain the task you want to accomplish. Finally, $(\boldsymbol{X}_t, \boldsymbol{\tau}_t, R_t, \boldsymbol{X}_{t+1})$ is stored in the experience pool M_t, and when the experience pool capacity is zero, the network is updated. In order to break the correlation between data, the stochastic sampling method was adopted to extract the sequences $m_t = (\boldsymbol{X}_t, \boldsymbol{\tau}_t, R_t, \boldsymbol{X}_{t+1}), i = 1, 2, \cdots, N$ from M_t for training.

The majority of model-free RL methods are based on policy iteration: policy evaluation and policy improvement [31]. The DDPG-based path planning algorithm also includes the above two steps: Critic to estimate state–action value function and Actor to increase the gradient of state–action value function.

We use an off-policy actor-critic network architecture, and the evaluated policy is not the same strategy as the improved policy, and the evaluated policy β is stochastic sampled from the memory M_t. Besides, since DDPG method is used, the policy probability $\pi(\boldsymbol{\tau}|\boldsymbol{X})$ will be replaced by the action corresponding to the deterministic policy $\mu(\boldsymbol{X})$. Therefore, Equation (19) which removes the integral over actions can be written:

$$\begin{aligned} J_\beta(\mu) &= \int_S \rho^\beta(\boldsymbol{X}) Q^\mu(\boldsymbol{X}, \mu(\boldsymbol{X})) d\boldsymbol{X} \\ &= \mathbb{E}_{\boldsymbol{X} \sim \rho^\beta}[Q^\mu(\boldsymbol{X}, \mu(\boldsymbol{X}))] \end{aligned} \tag{21}$$

Additionally, we use a differentiable state–action value function $Q^\omega(\boldsymbol{X}, \mu(\boldsymbol{X}))$ to replace the real state–action value function $Q^\mu(\boldsymbol{X}, \mu(\boldsymbol{X}))$ in Equation (19). So, the critic estimates the state–action value function $Q^\omega(\boldsymbol{X}, \mu(\boldsymbol{X})) \approx Q^\mu(\boldsymbol{X}, \mu(X))$, using an appropriate policy evaluation algorithm such as temporal-difference learning [32].

Critic-evaluation network parameter updated formula is as follows:

$$y_i = R_i + \gamma Q^{\omega'}(\boldsymbol{X}_{i+1}, \mu(\boldsymbol{X}_{i+1}|\theta^{\mu'})) \tag{22}$$

$$\delta_i = y_i - Q^\omega(X_i, \boldsymbol{\tau}_i) \tag{23}$$

$$L = \frac{1}{N}\sum \delta_i{}^2 \tag{24}$$

$$\omega_{t+1} = \omega_t + \alpha_\omega L \nabla_\omega Q^\omega(\boldsymbol{X}_t, \boldsymbol{\tau}_t) \tag{25}$$

where γ is discount factor, δ_i is TD-error, L is loss function and α_ω is critic-evaluation network learning rate.

Actor-evaluation network parameter updated formula is expressed:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum \nabla_{\theta^\mu}\mu(\boldsymbol{X}_i|\theta^\mu)\nabla_{\boldsymbol{t}} Q^\omega(X, \boldsymbol{\tau})|_{\boldsymbol{X}=\boldsymbol{X}_i, \boldsymbol{\tau}=\mu(\boldsymbol{X}_i)} \tag{26}$$

$$\theta^\mu_{t+1} = \theta^\mu_t + \alpha_{\theta^\mu}\nabla_{\theta^\mu} J \tag{27}$$

where α_{θ^μ} is actor-evaluation learning rate.

4. Reward function

There is also a key component of our system: action-judgement module which is a penalty function can only evaluate the output of the entire system. Different from supervised learning, supervised learning not only requires a large amount of preliminary data but also needs to calibrate and verify the data. Therefore, the reward function in the system is more evaluative than instructive, which makes it hard to be the correct structure. Further, the reward mechanism also implicitly explains the goal and process of the whole task, by optimising the path to increase reward constantly. By entering a state into the decision system, you can get immediate reward, which is different terms multiplied by a specific value rather than winning or losing in ordinary games. After the judgement is completed, the action is added to the control sequences because what we want is the global optimum of the control sequence and not the local optimum. In this system, the control sequence does not clearly indicate how long it is because the length of the sequence is determined by the complexity of the task. Consequently, the reward function can be expressed in the following parts.

First, our goal is to plan an optimal route to our destination while avoiding collision so the distance between the vessel and the target should be closer and closer. Put a negative sign in front of the distance term, indicating that the farther away from the end point, the less reward.

$$R_{\text{distance}} = -\lambda_{\text{distance}} d_{\text{tar}} \tag{28}$$

$$d_{\text{tar}} = \sqrt{\left(x - x_{\text{target}}\right)^2 + \left(y - y_{\text{target}}\right)^2} \tag{29}$$

where d_{tar} is the distance from the centre of target to the USV centre.

Second, supported r_o as the USV detection radius, d_i is the distance from the centre of obstacle to the USV centre and b_i is the ratio between d_i and the detection radius r_o. If the obstacle is within the detection range, the penalty for approaching the vessel increases exponentially, and thereby

$$R_{\text{collision}} = \begin{cases} -\lambda_{\text{collision}} \sum_{N_{\text{obs}}}^{i=1} \ln\left(\frac{1}{b_i}\right) & d_i < r_o \\ -g_{\text{collision}} & 0 \leq b_i < 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

$$d_i = \sqrt{(x - x_{obs_i})^2 + (y - y_{obs_i})^2} \tag{31}$$

$$b_i = \frac{d_i}{r_o} \tag{32}$$

where N_{obs} is the obstacle number in the detection range and (x_{obs_i}, y_{obs_i}) is the coordinates of the centre of the obstacles.

Similarly, the drifting phenomenon should be avoided as much as possible during the navigation process, so we should punish when the sway velocity v is greater than the surge velocity u. Thus, the drift term is defined as

$$R_{\text{drift}} = \begin{cases} -\lambda_{\text{drift}} e^{\frac{|v|}{|u|}} & |v| > |u| \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

The speed of vessel should be reduced at the end. This will ensure the safety of vessel and save energy.

$$R_{\text{end}} = \begin{cases} -\lambda_{\text{end}} e^{|u|+|v|} & d_{\text{tar}} < 2r_o \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

In the above equation, the relative values of different types of terms are important, but they are all multiplied by constant λ to balance the size of each term. There may be cases in which most of the penalties of vessel in the course of its voyage come from only one of them. Now, different types of terms are combined to form the following a heterogeneous avoidance reward function:

$$R(X_t) = \lambda^T R = \begin{bmatrix} \lambda_{\text{distance}} \\ \lambda_{\text{collision}} \\ \lambda_{\text{drift}} \\ \lambda_{\text{end}} \end{bmatrix}^T \begin{bmatrix} R_{\text{distance}}(x_t, y_t) \\ R_{\text{collision}}(x_t, y_t) \\ R_{\text{drift}}(u_t, v_t) \\ R_{\text{end}}(x_t, y_t, u_t, v_t) \end{bmatrix} \tag{35}$$

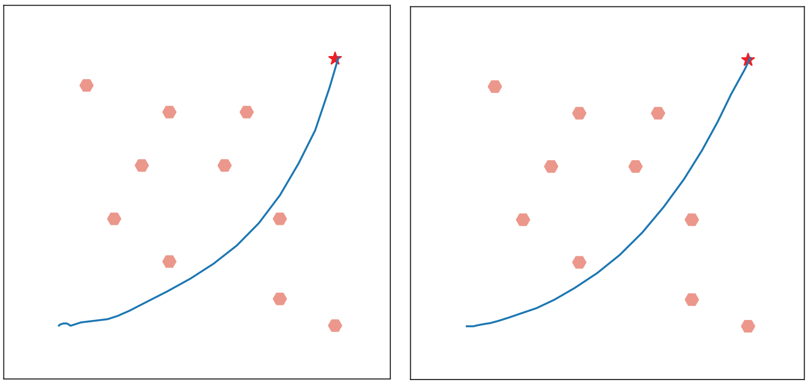
5. Simulation studies

In this section, the validity of the DDPG-based path planning algorithm will be demonstrated by simulation studies. The system uses the framework of the tensorflow [33] neural network. In all experiments, the same kinematics model which is unknown to the environment will be used. Additionally, the model parameters are implemented following the [19] in all cases. At the beginning of each epochs, v and $\boldsymbol{\eta}$ of the model are zero. The length of historical information to be considered T_b and the capacity of memory M_e and M_t is set to 6, 6, 10,000, respectively. In addition, the constants in the penalty equation of the USV voyage stage $\lambda_{\text{distance}}$, $\lambda_{\text{collision}}$, λ_{drift}, λ_{end} and $g_{\text{collision}}$ are 200, 30, 2, 6 and 1000, respectively. And the detection radius of the vessel r_o is set as 0.15.

We have the following symbols and colours to use in Figure 4: This rectangle is the navigational range of a vessel. The light red hexagon in the rectangle is obstacles in the sea. The green circle is the safe range of the vessel. The blue triangle in the green circle represents the vessel whose initial position in the lower left corner and end point is the red pentagram in the upper right corner. The vessel needs to plan an optimal route between the start and end points to avoid obstacles in unknown environments.

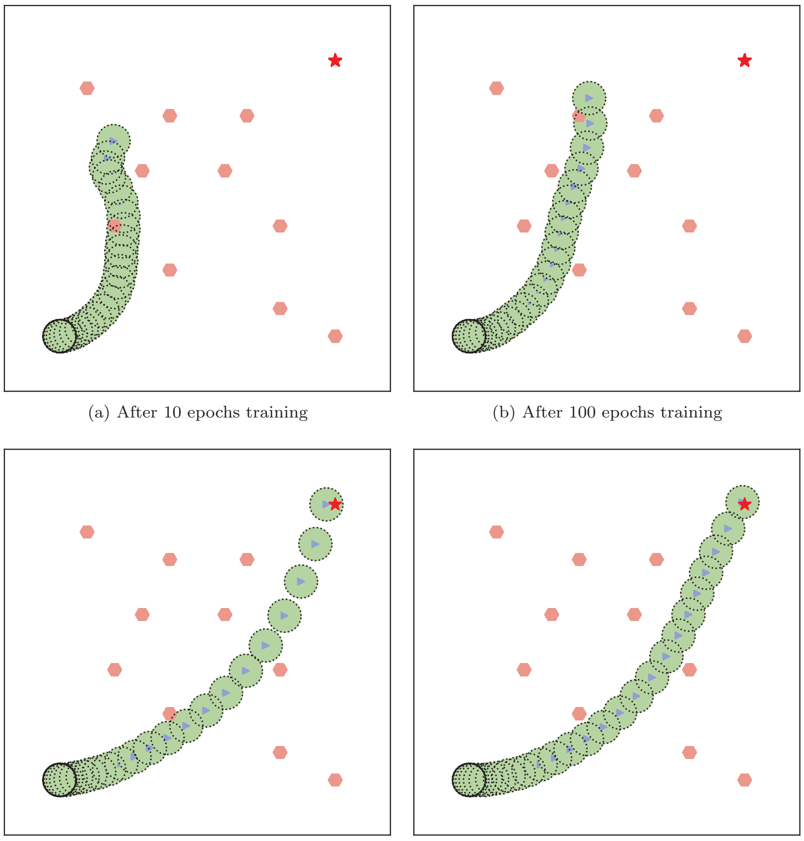
The training process of the DDPG-based algorithm which is shown in Appendix A is the same with and without disturbances. However, the USV trajectory will slightly wobble when a time-varying disturbance is added. It can be seen in Figure 3 which is the trajectory of USV centre point that the USV gets a beautiful path without disturbance within 5000 training epochs while, although it can find out the appropriate solutions within 5000 training epochs with disturbance, the trajectory is still a bit tortuous, especially near the initial state.

It can be seen from Figure 4(a) that after 10 epochs, the USV is still in the stage of exploration. Instead of heading towards NE, it heads towards the north and collides with obstacles. After 100 epochs (Figure 4(b)), the USV speed increased. At the same time, the trajectory of the USV has gradually begun to drift towards the destination under the influence of $R_{distance}$, and two



(a) Training with time-varyin disturbance (b) Training without time-varyin disturbance

Figure 3. The trajectory of USV centre point after training 5000 epochs.



(a) After 10 epochs training (b) After 100 epochs training

(c) After 1000 epochs training (d) After 5000 epochs training

Figure 4. The autonomous planning path during training with DDPG-based algorithm.

obstacles have been avoided. The USV can avoid obstacles to reach the target after 1000 epochs (Figure 4(c)), but it can be clearly seen that the speed is too fast to overshoot the destination. At this moment, R_{end} start to make an influence, which gives the USV enough time to slow down near the end. Therefore, the slower the destination, the greater the reward. In addition, since the R_{end} only works near the end point, the previous trajectory does not change much. The trajectory is shown in Figure 4(d).

6. Conclusions

In this paper, for path planning of an USV in unknown environments, the proposed DDPG-based method has been deployed as a decision system to make its control behaviour continuous. The proposed algorithm only needs the basic measures of the operational states and the detection information from obstacles. In addition, the penalty function can be freely designed according to the changes of different tasks and models with high usability and feasibility.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work is supported by the National Natural Science Foundation of P. R. China [under Grants 51009017 and 51379002]; the Fund for Dalian Distinguished Young Scholars [under Grant 2016RJ10]; the Stable Supporting Fund of Science and Technology on Underwater Vehicle Laboratory [SXJQR2018WDKT03]; and the Fundamental Research Funds for the Central Universities [under Grants 3132016314 and 3132018126].

References

[1] Li JL. Development and application of unmanned surface vehicle. Fire Control Comm Control. 2012;37(6):203–207.
[2] Swanepoel LH, Dalerum F, Hoven WV. Factors affecting location failure of GPS collars fitted to African leopards (*Panthera pardus*). South Afr J Wildl Res. 2010;40(1):10–15.
[3] Dantzig GB, Ramser JH. The truck dispatching problem. Manage Sci. 1959;6(1):80–91.
[4] Lozano-Pérez T, Wesley MA. An algorithm for planning collision-free paths among polyhedral obstacles. Commun ACM. 1979;22(10):560–570.
[5] Xu WB, Chen XB. Artificial moment method for swarm robot formation control. Sci China Inf Sci. 2008;51(10):1521–1531.
[6] Conn RA, Kam M. Robot motion planning on N-dimensional star worlds among moving obstacles. IEEE Transactions Robotics Automation. 2002;14(2):320–325.
[7] Mansor MA, Morris AS. Path planning in unknown environment with obstacles using virtual window. J Intell Robotic Syst. 1999;24(3):235–251.

[8] Sciavicco L, Siciliano B. A solution algorithm to the inverse kinematic problem for redundant manipulators. IEEE J Robotics Automation. 1988;4(4):403–410.

[9] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. Int J Robotics Res. 1986;5(1):90–98.

[10] Tu JP, Yang SX. Genetic algorithm based path planning for a mobile robot. IEEE International Conference on Robotics and Automation, Taipei. 2003, pp. 1221–1226.

[11] Trovato KI, Dorst L. Differential A*. IEEE Trans Knowl Data Eng. 2002;14(6):1218–1229.

[12] Hsieh HT, Chu CH. Improving optimization of tool path planning in 5-axis flank milling using advanced PSO algorithms. Robotics Comput Integr Manufacturing. 2013;29 (3):3–11.

[13] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013.

[14] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–533.

[15] Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. Nature. 2016;529(7587):484–489.

[16] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. Nature. 2017;550(7676):354–359.

[17] Cheng Y, Zhang WD. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing. 2018;272(10):63–73.

[18] Zhang B, Mao Z, Liu W, et al. Geometric reinforcement learning for path planning of UAVs. J Intell Robotic Syst. 2015;77(2):391–409.

[19] Wang N, Sun Z, Su SF, et al. Fuzzy uncertainty observer based path following control of underactuated marine vehicles with unmodelled dynamics and disturbances. International Conference on Fuzzy Theory and Its Applications; 2017; Pingtung. p. 1–6.

[20] Wang N, Sun Z, Zheng Z, et al. Finite-time sideslip observer-based adaptive fuzzy path-following control of underactuated marine vehicles with time-varying large sideslip. Int J Fuzzy Syst. 2018;20(6):1767–1778.

[21] Fossen TI. Handbook of marine craft hydrodynamics and motion control. Hoboken: John Wiley & Sons; 2011.

[22] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol. 1962;160(1):106–154.

[23] Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern. 1980;36(4):193–202.

[24] Lecun YL, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–2324.

[25] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on Machine Learning; 2010; Haifa. p. 807–814.

[26] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics; 2011; Fort Lauderdale. p. 315–323.

[27] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms. Proceedings of the 31th International Conference on Machine Learning; 2014; Beijing. p. 387–395.

[28] Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971. 2015.

[29] Wawrzynski P. Control policy with autocorrelated noise in reinforcement learning for robotics. Int J Machine Learn Comput. 2015;5(2):91–95.

[30] Uhlenbeck GE, Ornstein LS. On the theory of the Brownian motion. Phys Rev. 1930;36 (5):823–841.
[31] Arnold B. Reinforcement learning: an Introduction. Massachusetts: MIT press; 1998.
[32] Tsitsiklis JN, Roy BV. An analysis of temporal-difference learning with function approximation. IEEE Trans Automat Contr. 2002;42(5):674–690.
[33] Abadi M, Barham P, Chen J, et al. TensorFlow: a system for large-scale machine learning. Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation; 2016; Savannah. p. 265–283.

Appendix

Algorithm 1 DDPG-based path planning for USVs

```
1: Randomly initialise critic network $Q(X,\boldsymbol{\tau}|\omega)$ and $\mu(X_i|\theta^\mu)$ with weights $\omega$ and $\theta^\mu$
2: Initialise the target network $\omega'$ and $\theta^{\mu'}$ with weights $\omega' = \omega$ and $\theta^{\mu'} = \theta^\mu$
3: Initialise the $v_1 = \boldsymbol{\eta}_1 = [0 \quad 0 \quad 0]^T$ and $\boldsymbol{\tau}_u = \boldsymbol{\tau}_r = \sigma_1 = 0$
4: Initialise experience pool $M_t$ to capacity $N_t$
5: for $episode = 1, M$ do
6:    Initialise a random noise $\mathcal{N}$ for action exploration
7:    Initialise the historical observation buffer $M_e$ to capacity $T_b$
8:    Receive initial observation vector $s_1$
9:    for $t = 1, T$ do
10:      Store the initial observation $s_1$ in repository $M_e$
11:      Compress the current state $s_t = [\sigma_t \quad v_t \quad \boldsymbol{\eta}_t \quad \boldsymbol{\tau}_{t-1}]$
12:      if $t \leq T_b$, then
13:        Set $s_k = 0$, $k = T_b - t, \cdots, 0$
14:      end if
15:      Form the input state $X_t = [s_t \quad s_{t-1} \quad \cdots \quad s_{t-T_b}]^T$
16:      Select control behaviour $\boldsymbol{\tau}_t = \mu(X_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and
         exploration noise
17:      Execute $\boldsymbol{\tau}_t = [\boldsymbol{\tau}_u(t) \quad 0 \quad \boldsymbol{\tau}_r(t)]^T$ and observe $v_{t+1}$ and $\boldsymbol{\eta}_{t+1}$ and observe $X_{t+1}$
18:      Detect the obstacles $\sigma_{t+1}$
19:      Calculate the punishment $R_t$
20:      Store the new observation $s_t + 1$ in memory $M_e$
21:      Store transition $(X_t \quad \mathbf{t}_t \quad R_t \quad X_{t+1})$ in $M_t$
22:      if $M_t$ full, then
23:        Sample a random minibatch of $N$ transitions $(X_i \quad \boldsymbol{\tau}_i \quad R_i \quad X_{i+1})$ from $M_t$
24:        Set $y_i = R_i + \gamma Q^{\omega'}(X_{i+1}, \mu(X_{i+1}|\theta^{\mu'}))$
25:        Train the critic with $L = \frac{1}{N}\sum(y_i - Q^\omega(X_i, \boldsymbol{\tau}_i))^2$ as loss function
26:        Update the actor using the policy gradient:
             $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum(\nabla_{\theta^\mu}\mu(X_i|\theta^\mu)\nabla_{\boldsymbol{\tau}} Q^\omega(X,\boldsymbol{\tau})|_{X=X_i,\boldsymbol{\tau}=\mu(X_i)})$
27:        Update the target networks:
             $\theta^{\mu'} \leftarrow \varepsilon\theta^\mu + (1-\varepsilon)\theta^{\mu'}$
             $\omega' \leftarrow \varepsilon\omega + (1-\varepsilon)\omega'$
28:      end if
29:    end for
30: end for
```