IET Intelligent Transport Systems

Research Article

IET Journals
The Institution of Engineering and Technology

Motion control of unmanned underwater vehicles via deep imitation reinforcement learning algorithm

ISSN 1751-956X
Received on 6th May 2019
Revised 20th February 2020
Accepted on 16th March 2020
E-First on 1st May 2020
doi: 10.1049/iet-its.2019.0273
www.ietdl.org

Zhenzhong Chu[1], Bo Sun[1], Daqi Zhu[1] ✉, Mingjun Zhang[2], Chaomin Luo[3]

[1]*Shanghai Engineering Research Center of Intelligent Maritime Search & Rescue and Underwater Vehicles, Shanghai Maritime University, Shanghai 201306, People's Republic of China*

[2]*College of Mechanical and Electrical Engineering, Harbin Engineering University, Harbin 150001, People's Republic of China*

[3]*Department of Electrical and Computer Engineering, Mississippi State University, Mississippi MS 39762, USA*

✉ *E-mail: dqzhu@shmtu.edu.cn*

Abstract: In this study, a motion control algorithm based on deep imitation reinforcement learning is proposed for the unmanned underwater vehicles (UUVs). The algorithm is called imitation learning (IL) twin delay deep deterministic policy gradient (DDPG) (TD3). It combines IL with DDPG (TD3). In order to accelerate the training process of reinforcement learning, the supervised learning method is used in IL for behaviour cloning from the closed-loop control data. The deep reinforcement learning employs actor–critic architecture. The actor part executes the control strategy and the critic part evaluates current control strategy. The training efficiency of IL-TD3 is compared with DDPG and TD3. The simulation results show that the training results of IL-TD3 converge faster and the training process is more stable than both of them, the convergence rate of IL-TD3 algorithm during training is about double that of DDPG and TD3. The control performance via IL-TD3 is superior to PID in UUVs motion control tasks. The average track error of IL-TD3 is reduced by 70% than PID control. The average tracking error under thruster fault is almost the same as under normal condition.

1 Introduction

In recent years, unmanned underwater vehicles (UUVs) have been increasingly applied in submarine terrain exploration, military reconnaissance, and other fields. However, due to the complexity of the underwater environment, the precise motion control of UUVs is a difficult problem. At present, the control methods of UUVs have been used are PID control, adaptive control, fuzzy control, neural network control and several control methods combined [1, 2].

Most of the UUVs applied PID as control algorithm. The advantages of PID control algorithm are simple structure, easy implementation and high-reliability. The shortcoming is that when the underwater environment changes, such as current disturbance, the PID parameters need to be re-adjusted. Many researchers improved PID control algorithm and combined PID with other algorithm [3, 4]. There are also some UUVs used model-based control methods to tasks such as path following and trajectory tracking of UUVs [5, 6]. However, model-based control algorithms have requirements for the model accuracy of UUVs. If the model is not accurate, it often fails to perform well. With the development of deep learning technology and the improvement of computer performance, researchers tried to find a model-free intelligent algorithm to solve the problem of UUVs control.

Deep Q-Network (DQN) combines deep neural networks with traditional reinforcement learning algorithms and shows good results on many issues [7]. However, DQN cannot be used for continuous motion space control, then deep deterministic policy gradient (DDPG) was proposed [8]. The DDPG combined the deterministic policy gradient (DPG) with the actor–critic architecture to solve the problem [9]. DDPG can achieve great performance sometimes, but it is frequently brittle concerning hyperparameters and easy to overestimate current policy. The performance is not ideal in complex environments. Therefore, the twin delay DDPG algorithm (TD3) based on DDPG was presented [10]. It successfully solved the problems of DDPG, and achieved good results in the MuJoCo simulation environment [11]. However, deep reinforcement learning (DRL) requires a large amount of data, and agents often need many explorations to find the right strategy. Many researchers have used expert demonstrations for DRL to speed up the training process and achieved superb results [12, 13].

Some researchers proposed algorithms based on DRL in the UUVs control. Cui *et al.* [14] applied DRL to implement the adaptive neural network control of UUVs trajectory tracking. Carlucho *et al.* [15] developed a DRL method for adaptive low-level control of UUVs. Wu *et al* [16]. proposed a model-free DRL algorithm of UUVs depth control. Preliminary research results showed that DRL is feasible in the UUVs control and can achieve a good motion control performance.

In this work, a model-free DRL algorithm is modified for motion control of UUVs with the imitation learning (IL) method. The IL method makes up for the shortcomings in the model-free DRL algorithm. The control data given by PID algorithm are collected as expert demonstrations for pre-training by supervised learning. The parameters of policy part in DRL are initialised in this way. The TD3 algorithm is used to achieve self-enhancement based on PID control strategy. It compensates for the approximation error in actor–critic methods. The motion control tasks of UUVs are better completed by the IL-TD3 algorithm than PID and other pure DRL methods. The innovation of this paper is mainly reflected in the following two aspects: (i) the proposed algorithm combines IL with DDPG (TD3). (ii) Combining the proposed algorithm with the motion control of UUVs, a new motion control algorithm for UUVs is proposed. The rest of the paper is structured as follows: Section 2 presents the dynamic model of UUVs. Section 3 describes the deep IL control algorithm. The simulation and results analysis are presented in Section 4. The summary is given in Section 5.

2 Dynamics of UUVs

In this section, the dynamic model of UUVs is presented. In general, two reference systems are defined, namely the body-fixed frame and the inertial frame. The position and attitude of the UUV in inertial frame are represented by vector $\boldsymbol{\eta} = [x \quad y \quad z \quad \varphi \quad \theta \quad \psi]^{\mathrm{T}}$, $x, y, z \in R$ denote the Cartesian position of the UUV, and $\varphi, \theta, \psi \in R$ denote the attitude of roll,

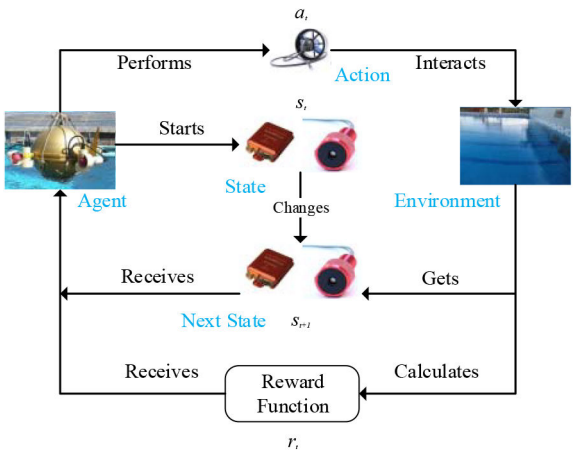

Fig. 1 *Reinforcement learning model of UUV*

pitch and yaw, respectively. The vector $\boldsymbol{\nu} = [u \quad v \quad w \quad p \quad q \quad r]^{\mathrm{T}}$ represents the linear and angular velocities of UUVs in the body-fixed frame, $u,\ v,\ w \in R$ represent the surge, sway and heave velocities, respectively, $p,\ q,\ r \in R$ represent the roll, pitch and yaw angular velocities, respectively. The relation between the vector in the inertial frame and the vector in the body-fixed frame is [17]

$$\dot{\boldsymbol{\eta}} = J(\boldsymbol{\eta})\boldsymbol{\nu} \tag{1}$$

where $J(\boldsymbol{\eta}) = [J_1, 0_{3x3}; 0_{3x3}, J_2]$ and (see (2)) , where $c(\cdot)$, $s(\cdot)$ and $t(\cdot)$ imply $\cos(\cdot)$, $\sin(\cdot)$ and $\tan(\cdot)$, respectively.

The dynamic model of UUVs is given by

$$[\boldsymbol{M}_{\mathrm{RB}} + \boldsymbol{M}_{\mathrm{A}}]\dot{\boldsymbol{\nu}} + [\boldsymbol{C}_{\mathrm{RB}} + \boldsymbol{C}_{\mathrm{A}}]\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \tau \tag{3}$$

where $\boldsymbol{M} = \boldsymbol{M}_{\mathrm{RB}} + \boldsymbol{M}_{\mathrm{A}}$ is the inertia matrix, $\boldsymbol{M}_{\mathrm{RB}}$ represents the rigid body mass matrix and $\boldsymbol{M}_{\mathrm{A}}$ represents the mass matrix of additional terms. $\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{\mathrm{RB}} + \boldsymbol{C}_{\mathrm{A}}$ is a Coriolis term and centrifugal matrix, composed of the rigid body parts $\boldsymbol{C}_{\mathrm{RB}}$ and the additional mass parts $\boldsymbol{C}_{\mathrm{A}}$. $\boldsymbol{D}(\nu)$ is hydrodynamic loss term matrix, $\boldsymbol{g}(\boldsymbol{\eta})$ is the static (gravity and buoyancy) term matrix. $\boldsymbol{\tau} = [X \quad Y \quad Z \quad K \quad M \quad N]^{\mathrm{T}}$ is the force and torque vector in the body-fixed frame.

It is assumed that the UUV has n thrusters. They can generate three horizontal and three rotational forces and torque, such that the UUV can complete 6 degrees of freedom motion. Therefore, $\boldsymbol{\tau}$ can be described as $\boldsymbol{\tau} = \boldsymbol{E}_{6\times n}\boldsymbol{F}_{n\times 1}$, where $\boldsymbol{E}_{6\times n}$ is the layout matrix of the n thrusters and $\boldsymbol{F}_{n\times 1}$ is the vector represents the force of each thruster.

3 Previous researches background

In this section, the basics of reinforcement learning and IL are introduced. These consists the fundamental of deep imitation reinforcement learning (DIRL) control design for UUVs, which will be introduced in the next sections.

3.1 Reinforcement learning statement

Reinforcement learning refers to the process in which the agent gradually learns the optimal strategy according to the reward signal in the constant interaction with the environment. Unlike supervised learning, agents do not have an explicit action specification, but instead, try to find out which behaviours can produce the greatest reward by trying different strategies.

The standard model of reinforcement learning is shown in Fig. 1. At each time step t, the agent performs the action $a_t \in A$ in state $s_t \in S$. Where A represents a set of control actions. S presents a set of all possible states. It receives the reward r_t and the next state s_{t+1}[18]. In the case of UUVs, a_t refers to the motion of UUVs, such as the thrust of thrusters. s_t refers to the states of UUVs, such as position, the linear and angular velocities of UUVs, these states are measured by sensory system. The reward r_t is used to measure the difference between the states of the UUVs and the target values. The policy π is a state-to-action mapping, which refers to a distribution on the UUV motion set for the state of UUV and $\pi(a|s) = p[a_t = a|s_t = s]$. The state-action value function when carrying out a in s following π is [7]

$$Q_\pi(s,a) = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle| s_t = s, a_t = a\right] \tag{4}$$

where $\gamma \in (0,1)$ is a discount factor used to calculate the cumulative return. The UUV's goal is to find the optimal policy π_μ to achieve desired state from the beginning of distribution $J(\mu) = \mathrm{E}_{s_i\sim\rho_\pi, a_i\sim\pi}[R_0]$ with the parameters μ.

3.2 Actor–critic architecture

The proposed DIRL algorithm employs an actor–critic method. The actor–critic method is a merger of value-based and policy-based reinforcement learning algorithms [19]. As shown in Fig. 2, two neural networks are used to implement this method. Fig. 2*a* is the policy gradient network (actor). The input of this network is state s_t, and the output is action a_t, then it can get the next state s_{t+1}. Fig. 2*b* is the state-action value network (critic). The input of this network is the state s_t and action a_t, and the output is the state-action value $Q_\pi(s_t, a_t)$ under policy π. The value function under policy π is recursively defined by the bellman equation as [7]

$$\mathscr{V}_\pi(s) = \mathrm{E}_\pi\{r_t + \gamma\mathscr{V}_\pi(s_{t+1})|s_t = s\} \tag{5}$$

$$J_1(\eta) = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\varphi) - s(\psi)c(\varphi) & c(\psi)s(\theta)c(\varphi) + s(\psi)s(\varphi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\varphi) + c(\psi)c(\varphi) & s(\psi)s(\theta)c(\varphi) - c(\psi)s(\varphi) \\ -s(\theta) & c(\theta)s(\varphi) & c(\theta)c(\varphi) \end{bmatrix}$$

$$J_2(\eta) = \begin{bmatrix} 1 & t(\theta)s(\varphi) & c(\varphi)t(\theta) \\ 0 & c(\varphi) & -s(\varphi) \\ 0 & \dfrac{s(\varphi)}{c(\theta)} & \dfrac{c(\varphi)}{c(\theta)} \end{bmatrix} \tag{2}$$

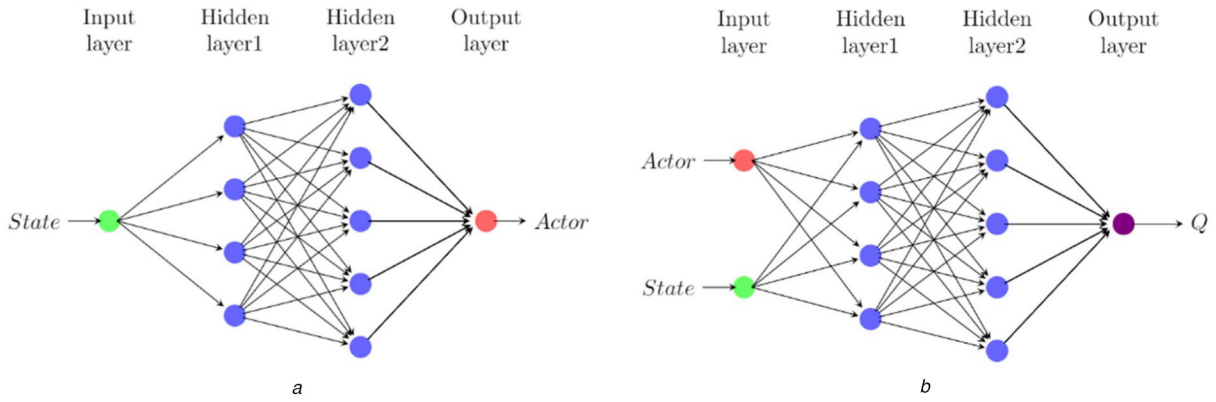

Fig. 2 *Structure of the actor and critic networks*
(a) Actor network, ***(b)*** Critic network

The value of $\mathscr{V}_\pi(s)$ is obtained by the critic network, with s_t and a_t as input. The temporal difference error δ_t of state is calculated by [7]

$$\delta_t = Q_\pi(s_t, a_t) - \mathscr{V}_\pi(s_t) \tag{6}$$

Then, the weights ω of critic network and the weights θ of actor network are updated by [9]

$$\omega \leftarrow \omega + \beta\delta_t \nabla_\omega \mathscr{V}_\pi(s_t, \omega) \tag{7}$$

$$\theta \leftarrow \theta + \alpha\delta_t \nabla_\theta \log \pi(a_t|s_t, \theta) \tag{8}$$

The symbol ∇ represents calculation of the gradient, where α and β are the learning rates of the actor and critic, respectively.

Although the actor–critic algorithm has the advantage that it updates faster than the traditional policy gradient method, there is a shortcoming, that is the algorithm depends on the critic's value judgment. In order to solve this problem, DDPG algorithm was proposed. DDPG used an actor–critic approach on the DPG algorithm in the actor section. Where DPG is a policy-gradient method, and the state-action value function of DPG is [9]

$$Q_\omega(s_t, a_t) = \mathrm{E}[r(s_t, a_t) + \gamma Q_\omega(s_{t+1}, \mu(s_{t+1}))] \tag{9}$$

where $\mu(s_{t+1})$ represents the actor network output with the input state s_{t+1}. DPG uses off-policy method to update the parameters of Q function. It updates the policy parameter θ by [9]

$$\nabla_\theta J_\beta(\theta) = \mathrm{E}_{s\sim\rho^\beta}\left[\nabla_a Q_\omega(s, a)|_{s=s_t, a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s)|_{s=s_t}\right] \tag{10}$$

where $Q_\omega(s_t, a_t)$ is the Q function, $\pi_\theta(s_t)$ represents deterministic strategy, ρ^β is the probability distribution function of states, and $J(\theta)$ is the performance function.

The DDPG adopts two deep neural networks to approximate $Q_\omega(s, a)$ and $\pi_\theta(s)$, together with the ideas of the replay buffer and target networks. Replay buffer is applied to break the Markov nature between the sampled data, and the two target networks are used to ensure the stability of the training process.

Although DDPG can usually achieve good performance, it is often vulnerable to hyperparameters and other kinds of tuning [20]. DDPG algorithm TD3 adds three tricks to solve these problems. The first one is double critic networks, the second one is target policy smoothing, and the last but not least is policy delays.

In this paper, the input of the actor network is the UUV's sensor states, the output of the actor network is the UUV's thrusters' output. The input of the critic network are the sensor states and thrusters' output of the UUV, the output is the state-action value Q.

3.3 Deep IL

DRL algorithms based on actor–critic method have shown great performance in the simulation environments. However, considering DRL is a model-free method, it requires a large amount of training data, and it has high trial and error cost in the real environment, which is difficult to be directly applied to UUVs motion control.

In order to solve this problem, the method DIRL combing IL and DRL is proposed to accelerate the training process of DRL and maintain the stability of the UUVs during training. In the IL part, the existing expert strategy trajectory is used to quickly initialise the model and then reinforcement learning method is applied to explore and feedback the environmental state.

IL refers to learning from the demonstrations provided by the expert. The simplest way to implement IL is supervised learning. Decision data from human experts are generally provided as $\{\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_m\}$. Each decision contains a sequence of states and actions $\mathscr{P}_i = s_1^i, a_1^i, s_2^i, a_2^i, \ldots, a_n^i$. It abstracts the state-action pairs to construct a new set $\mathscr{D} = \{(s_1, a_1), (s_2, a_2), (s_3, a_3), \ldots\}$. The action strategy made by the expert for each state is used as the sample label for supervised learning. This method is called behaviour cloning. The IL part is used as pre-training, the combination of DRL is called DIRL.

4 DIRL control design for UUVs

In this section, the motion control of UUVs and previous researches are combined, the framework of DIRL control and detailed steps are introduced. The motion control of UUV is a continuous control task, so the TD3 based on actor–critic architecture is adopted. The state of UUV are consist of position vector $\boldsymbol{\eta}$ and velocity vector $\boldsymbol{\nu}$, the command of UUV is thrust vector $\boldsymbol{F}$.

4.1 Framework of DIRL control

The proposed DIRL control algorithm is IL-TD3 (IL-TD3), which is based on TD3 algorithm. Fig. 3 is the overall framework, where the 'actor' and 'critic' represent actor network and critic network, respectively. Six independent deep neural networks are adopted to implement TD3.

As can be seen in the framework, the raw data of UUV are obtained from the sensory system. UUV's state vector $\boldsymbol{s}_t = [x_t \;\; y_t \;\; z_t \;\; \varphi_t \;\; \theta_t \;\; \psi_t \;\; u_t \;\; v_t \;\; w_t \;\; p_t \;\; q_t \;\; r_t]^{\mathrm{T}}$. The UUV is assumed to have eight thrusters, the thrust vector $\boldsymbol{F}_t = [F_{1t} \;\; F_{2t} \;\; F_{3t} \;\; F_{4t} \;\; F_{5t} \;\; F_{6t} \;\; F_{7t} \;\; F_{8t}]$. First, the PID control algorithm is used to control the UUV. The formula of PID algorithm is [3]

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\mathrm{d}t + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{11}$$

In IL part, vectors $\boldsymbol{s}_t$ and $\boldsymbol{F}_t$ during the PID control processes are abstracted to construct a set $\{(s_1, F_1), (s_2, F_2), \ldots\}$ as expert demonstration data. Then these data are randomly sampled for pre-training actor network with supervised learning. After that the behaviour cloning is completed. In the RL part, the training progress is similar as that in TD3.

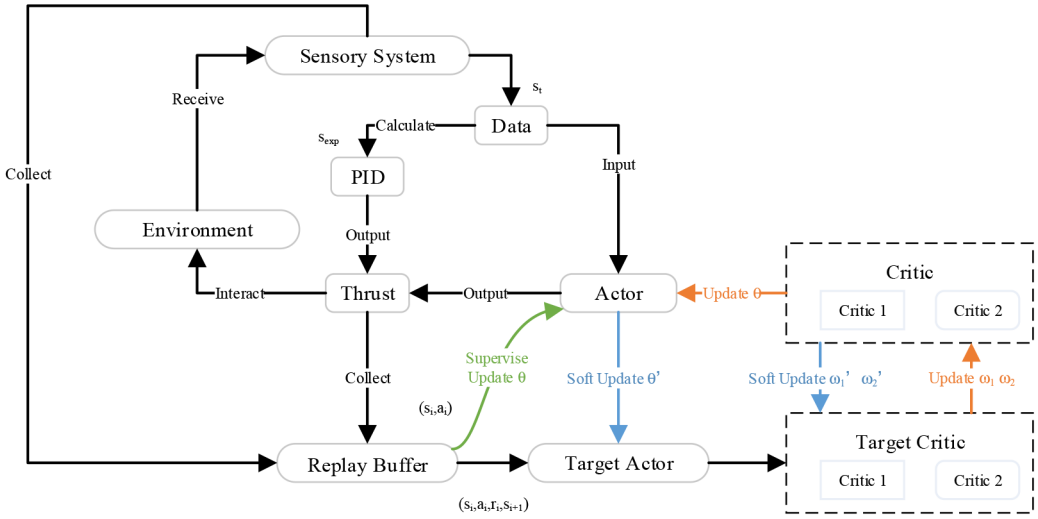

Fig. 3 *Overview of the control system*

4.2 Specific steps applied to UUVs

Algorithm 1 is the pseudocode of the proposed algorithm. The detailed steps of DIRL control design for UUVs are outlined (Fig. 4).

As can be seen in the first line of Algorithm 1, the hyperparameters need to be set before training are shown in Table 1.

In Table 1, actor-lr 1, actor-lr 2 and critic-lr denote the learning rate of actor network in IL, the learning rate of actor network in DRL and the critic network learning rate in DRL, respectively. soft-lr represents the soft update rate. The size1 and size2 denotes the sample size. Iteration 1 and iteration 2 represents the number of iterations in IL and RL, respectively.

By randomly initialise the weights ω_1, ω_2 and θ of the critic networks and the actor network, the corresponding weights ω_1', ω_2' and θ' are assigned to the target critic networks and the target actor network. At the beginning of each training loop, the exploration noise is initialised, the expected state $S_{\exp}$ and the initial state S_1 are randomly initialised.

From line 9 to line 34 is the inner loops of each episode. The current state S_t is taken as the input of the actor network. The action is selected according to the amount of data in $\mathscr{RB}$. When $|\mathscr{RB}| < m_{\max}$, it outputs the result of PID algorithm. Otherwise, it acts from the actor network with random noise

$$\boldsymbol{F}_t = \pi_\theta(S_t) + \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{12}$$

where $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{t^2}{2\sigma^2}\right)$ is Gauss noise with parameter $\mathcal{N}(0,\sigma^2)$.

The reward r_t is calculated based on the state s_t and action F_t at time t, then it gets the state s_{t+1} of the next moment. (s_t, F_t, r_t, s_{t+1}) is the state transition tuple. When $|\mathscr{RB}| < m_{\max}$, the data are obtained from PID control, and a batch of experience tuples (s_i, F_i) are sampled with size K from $\mathscr{RB}$. The state s_t is taken as input and action F_t is used as label to train the actor network with supervised learning. The loss function is defined as

$$L(\theta) = \frac{1}{2K}\sum_{i=1}^{K}(F_i - \pi_\theta(s_i))^2 + 0.5 \parallel \theta \parallel_2^2 \tag{13}$$

Regular terms $\parallel \theta \parallel_2^2$ are used to prevent overfitting. The coefficient of the regular term is set to 0.5. Thus,the gradient of loss function is

$$\nabla_\theta L(\theta) = -\frac{1}{K}\sum_{i=1}^{K}(F_i - \pi_\theta(s_i))\frac{\partial \pi_\theta(s_i)}{\partial \theta} + \theta \tag{14}$$

The weights of the actor network are updated by the random gradient descent method with $\mathscr{T}$ times per episode.

When $|\mathscr{RB}| \geq m_{\max}$, N state transition tuples $(s_i, F_i, r_i, s_{i+1})_{i=1\sim N}$ are randomly sampled for mini-batch training. s_{i+1} in the random sampled data is taken as the input of the target actor network. The Gauss noise $\frac{1}{\tilde{\sigma}\sqrt{2\pi}}\exp\left(-\frac{t^2}{2\tilde{\sigma}^2}\right)$ with scale c and parameter $\mathcal{N}(0,\tilde{\sigma}^2)$ is added on the output F_i to get F_i' as

$$F_i' = \pi_{\theta'}(s_{i+1}) + \frac{1}{\tilde{\sigma}c\sqrt{2\pi}}\exp\left(-\frac{t^2}{2\tilde{\sigma}^2}\right) \tag{15}$$

The Q-value label y_i is calculated by

$$y_i = r_i + \gamma \min_{j=1,2} Q_{\omega'_j}(s_{i+1}, F'_i) \tag{16}$$

The loss functions of critic networks are defined as

$$L(\omega_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{\omega_1}(s_i, F_i)\right)^2 \tag{17}$$

$$L(\omega_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{\omega_2}(s_i, F_i)\right)^2 \tag{18}$$

After that, the gradient of the loss functions is

$$\nabla_{\omega_1} L(\omega_1) = -\frac{2}{N}\sum_{i=1}^{N}\left(y_i - Q_{\omega_1}(s_i, F_i)\right)\frac{\partial Q_{\omega_1}(s_i, F_i)}{\partial \omega_1} \tag{19}$$

$$\nabla_{\omega_2} L(\omega_2) = -\frac{2}{N}\sum_{i=1}^{N}\left(y_i - Q_{\omega_2}(s_i, F_i)\right)\frac{\partial Q_{\omega_2}(s_i, F_i)}{\partial \omega_2}. \tag{20}$$

The 'delay' method is applied in the policy part. That is, in each episode, the critic part is updated $\mathscr{I}$ times, while the actor part $\mathscr{I}/f$ times. The gradient used to update the actor network is

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\nabla_a Q_{\omega_1}(s, F)\Big|_{s=s_i, F=\pi_\theta(s_i)}\nabla_\theta \pi_\theta(s)\Big|_{s=s_i}. \tag{21}$$

Adam is employed to update the weights of the actor and critic networks [21]. Finally, target network is updated with soft update method as

$$\omega'_i \leftarrow (1-\xi)\omega'_i + \xi\omega_i \quad i = 1, 2 \tag{22}$$

$$\theta' \leftarrow (1-\xi)\theta' + \xi\theta \tag{23}$$

1: **Inputs:** $T, M, K, N, \mathcal{T}, \mathcal{I}, f, \sigma, \tilde{\sigma}, c, m_{min}, m_{max}, \alpha, \beta, \lambda, \xi, \gamma$

2: Initialize critic networks $Q_{\omega_1}, Q_{\omega_2}$ and actor network π_θ with random parameters ω_1, ω_2, θ

3: Initialize target networks with parameters $\omega_1' \leftarrow \omega_1$, $\omega_2' \leftarrow \omega_2$, $\theta' \leftarrow \theta$

4: Initialize replay buffer $\mathcal{RB}$

5: **For** $t = 1$ **to** T **do**

6: Set desired state s_{exp}

7: Initialize exploration noise with parameter $\mathcal{N}(0, \sigma^2)$

8: Receive initial state s_1 from the sensory system

9: **For** $j = 1$ **to** M **do**

10: **If** $|\mathcal{RB}| \geq m_{max}$ **then**

11: Execute action $F_t = \pi_\theta + \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{t^2}{2\sigma^2}\right)$

12: **Else**

13: Execute F_t from PID algorithm

14: **End if**

15: Calculate the reward r_t, receive the next state s_{t+1}

16: Store transition tuple (s_t, F_t, r_t, s_{t+1}) in $\mathcal{RB}$

17: **If** $m_{min} \leq |\mathcal{RB}| < m_{max}$ **then**

18: Sample K transitions $(s_i, F_i)_{i=1\sim K}$ from $\mathcal{RB}$

19: Update actor network π_θ $\mathcal{T}$ times

20: **End if**

21: **If** $|\mathcal{RB}| \geq m_{max}$ **then**

22: Sample a random mini-batch of N transitions $(s_i, F_i, r_i, s_{i+1})_{i=1\sim N}$ from $\mathcal{RB}$

23: **For** $i = 1$ **to** N **do**

24: Obtain F_i' from target actor network

25: Calculate y_i from target critic networks

26: **End for**

27: Update critic networks $Q_{\omega_1}, Q_{\omega_2}$ $\mathcal{I}$ times

28: **If** $\mathcal{I}\ mod\ f$ **then**

29: Update actor network π_θ by the deterministic policy gradient

30: Soft update target networks

31: **End if**

32: Discard the oldest samples

33: **End if**

34: **End for**

35: **End for**

Fig. 4 *Algorithm 1: DIRL algorithm for UUV motion control*

Table 1 Hyperparameters for training

Parameter	Symbol	Parameter	Symbol
episodes	T	noise1	σ
steps	M	noise2	$\tilde{\sigma}$
size1	K	noise2 scale	c
size2	N	actor-lr 1	λ
iterations 1	$\mathcal{T}$	actor-lr 2	α
iterations 2	$\mathcal{I}$	critic-lr	β
frequency	f	soft -lr	ξ
storage limit	$m_{\max}, m_{\min}$	discount	γ

Table 2 Dynamic parameters

Parameters	Value	Parameters	Value
m	125.0	$Z_{w\|w\|}$	148
$Z_{\dot{w}}$	62.39	Z_w	100
I_{xx}	4.629	$K_{p\|p\|}$	280
I_{yy}	4.629	K_p	230
$K_{\dot{p}}$	0	$M_{q\|q\|}$	280
$M_{\dot{q}}$	0	M_q	230
B	1224.17	W	1226.25
R	0.381	s	$\sin(1/4\pi)$

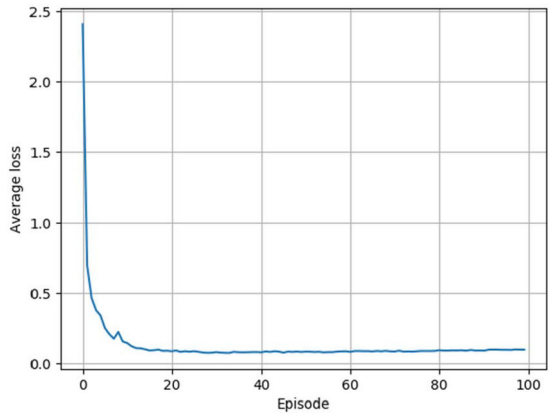

Fig. 5 *Average loss in IL for constant depth and attitude control*

when the amount of data in replay buffer $\mathcal{RB}$ reaches its storage limit, the oldest tuple is removed to keep the training data more meaningful.

5 Simulations

In order to verify the superiority of the proposed IL-TD3 algorithm, the simulations are carried out in this section.

ODIN UUV with four vertical thrusters and four horizontal thrusters is applied as simulation model. In simulation, the control tasks only involve heave, pitch and roll motions in the three degrees of freedom, that is, only four vertical thrusters are used. The simplified thruster layout matrix $\boldsymbol{E}_{3\times4}$ is

$$\boldsymbol{E} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ Rs & Rs & -Rs & -Rs \\ Rs & -Rs & -Rs & Rs \end{bmatrix} \tag{24}$$

where $R$is the distance between ODIN's centre and the centres of each thruster, $s$denotes the angle between the straight line (through the thruster's centre and the ODIN's centre) and the horizontal line.

The ODIN dynamic equations are as follows [22]:

$$\begin{cases} Z = (B-W)\cos\theta\cos\varphi + (m+Z_{\dot{w}})\dot{w} + Z_w w + Z_{w|w|}w|w| \\ K = I_{xx}\dot{p} + K_p p + K_{\dot{p}}\dot{p} + K_{p|p|}|p|p \\ M = I_{yy}\dot{q} + M_q q + M_{\dot{q}}\dot{q} + M_{q|q|}|q|q \\ \dot{z} = \cos\theta\cos\varphi w \\ \dot{\varphi} = p + q\tan\theta\sin\varphi \\ \dot{\theta} = \cos\varphi q \end{cases} \tag{25}$$

The dynamic parameters of ODIN are given in Table 2 [22]. In Table 2, m represents the mass of the UUV; W and B represents the UUV's weight and buoyancy force; the inertia moment in x and y directions are represented by I_{xx} and I_{yy}. $Z_{\dot{w}}$, $K_{\dot{p}}$ and $M_{\dot{q}}$ denote the hydrodynamic added mass coefficients of the UUV; Z_w, K_p and M_q represents the body lift force and moment coefficients; $Z_{w|w|}w$, $K_{p|p|}$ and $M_{q|q|}$ are the cross-flow drag coefficients.

The deep learning framework PyTorch is used for training under Ubuntu system. All training processes are done on the computer with an i7-8750H processor and an NVIDIA GeForce GTX 1060 GPU. Dynamic model of ODIN is set up by Python libraries Scipy and Sympy. In simulations, deep full connection networks are used to implement both the actor part and the critic part. For the actor network, the input are the states of UUV, and three hidden layers are used, and the numbers of neurons are 300, 200 and 100, respectively. ReLU is employed in hidden layers as the activation function. Tanh is applied as activation function in the output layer to regularise the actions to $[-1, 1]$. The input of critic networks are the states and actions of the UUV, and the numbers of neurons of the three hidden layers are 400, 300 and 200, respectively to ensure the training effect. The activation function of hidden layers is ReLU, to avoid gradient disappearance. The output of critic part is Q value. The target networks adopted the same structure of actor and critic networks.

5.1 Control task 1: constant depth and attitude control

In this control task, the state vector $\boldsymbol{s}_t = \begin{bmatrix} z_t & \varphi_t & \theta_t & w_t & p_t & q_t \end{bmatrix}^{\mathrm{T}}$ for the depth, roll angles and pitch angles and the corresponding velocities information of the UUV is obtained from the sensory system. $\boldsymbol{F}_t = \begin{bmatrix} F_{1_t} & F_{2_t} & F_{3_t} & F_{4_t} \end{bmatrix}^{\mathrm{T}}$ is the thrust vector of four thrusters in the vertical direction at time t. The thrust of each thruster is limited to $[-60N, 60N]$. Before training, the hyperparameters in Table 1 are set as

$$\begin{aligned} &T = 400, M = 600, K = 64, N = 100, \\ &\mathscr{T} = 10, \mathscr{I} = 4, f = 2, \sigma = 0.1, \tilde{\sigma} = 0.3, c = 0.5, \\ &m_{\min} = 60000, m_{\max} = 120000, \alpha = \beta = 0.001, \\ &\lambda = 0.001, \gamma = 0.99, \xi = 0.005 \end{aligned} \tag{26}$$

These parameters are confirmed after many experiments, and good training effects can be obtained under these parameters. The reward function is designed as (see (27)) . The expected state vector is set as $\boldsymbol{s}_{\exp} = \begin{bmatrix} z_{\exp} & \varphi_{\exp} & \theta_{\exp} & w_{\exp} & p_{\exp} & q_{\exp} \end{bmatrix}^{\mathrm{T}}$. $d(z_t)$, $d(\varphi)$ and $d(\theta_t)$ are the absolute value of the errors between the expected values and the actual values. $\boldsymbol{\Lambda} = \mathrm{diag}[1.5 \quad 5.0 \quad 5.0]$ is the diagonal matrix used to weight the importance of each state variable. $\sum_{i=1}^{4}|F_{it}|$ is the sum of the absolute thrust values of the thrusters. $\| F_{t-1} - F_t \|$ is set to prevent thruster thrust mutation. Case 1 represents $d(z_t) \le 0.05$, $d(\varphi_t) \le 0.01$ and $d(\theta_t) \le 0.01$. Case 2 represents $d(z_t) \ge 4.5$, $d(\varphi_t) \ge 0.2$ and $d(\theta_t) \ge 0.2$. During training, if $r_t = -100$, the current training episode is terminated early. Before the RL, the supervised learning method is used to pre-train the actor network, and the curves of the average loss in per episode are shown in Fig. 5. The loss function in the IL part goes down with the training episodes increase, changed from about 2.5 to 0.2. After 100 episodes training, the behaviour cloning is almost complete. Average rewards curves with 300 training episodes of each algorithm are shown in Fig. 6. It can be seen that the least reward TD3 and DDPG achieved is -43 and -39, respectively,

$$r_t = \begin{cases} -0.01\sum_{i=1}^{4}|F_{it}| - 0.1\left(w_t^2 + p_t^2 + q_t^2\right) - 0.001 \| F_{t-1} - F_t \| & \text{case 1} \\ -100 & \text{case 2} \\ -\left(\boldsymbol{s}_t - \boldsymbol{s}_{\exp}\right)\Lambda\left(\boldsymbol{s}_t - \boldsymbol{s}_{\exp}\right)^{\mathrm{T}} - 0.01\sum_{i=1}^{4}|F_{it}| - 0.1\left(w_t^2 + p_t^2 + q_t^2\right) - 0.001 \| F_{t-1} - F_t \| & \text{others} \end{cases} \tag{27}$$

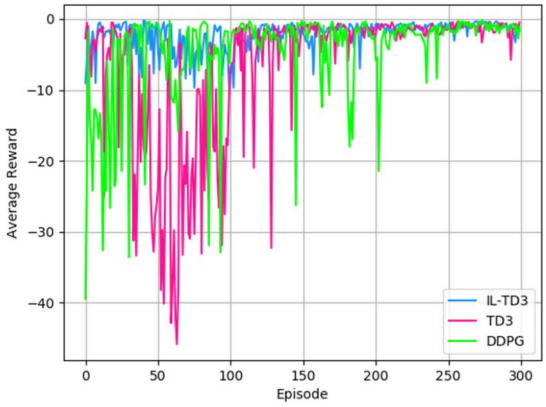

Fig. 6 *Average return reward curves in DRL for constant depth and attitude control*

Table 3 Sampled rewards of three algorithms in constant depth and attitude control

Episodes	IL-TD3	TD3	DDPG
1–30	−9.05	−2.76	−39.45
31–60	−1.03	−1.49	−33.59
61–90	−5.60	−37.69	−11.54
91–120	−1.37	−9.86	−4.98
121–150	−1.89	−4.13	−3.77
151–180	−2.09	−1.28	−0.96
181–210	−1.66	−1.64	−1.50
211–240	−2.44	−1.27	−2.10
241–270	−1.50	−1.35	−1.04
271–300	−0.50	−0.99	−0.86

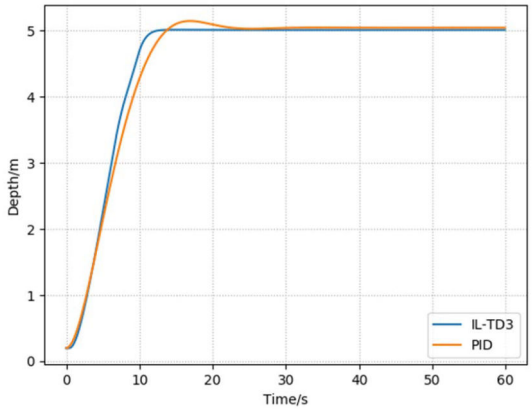

Fig. 7 *Depth by PID and IL-TD3 without current disturbance, initial depth is 0.2 m*

while the least reward IL-TD3 achieved is about −9. During the training process, after about 200 episodes, the IL-TD3 algorithm can stabilise about the maximum reward, while TD3 and DDPG still have drastic changes. It can be concluded that the IL-TD3 algorithm converges faster and more stable during training constant depth and attitude control of UUV.

The average rewards of three algorithms during the 300 training episodes are sampled per 30 episodes. The sampled rewards are shown in Table 3, it's obviously that the IL-TD3 achieved the best performance during the training process, and the DDPG performed worst. The reward of DDPG is least in all comparison data.

After completing the training, trained model of IL-TD3 is tested under normal conditions and with current disturbance, respectively. Then, the contrast simulations of motion control are carried out with PID control algorithm.

Three PID for depth control, roll angle control and pitch angle control are applied, respectively. (K_p, K_i, K_d) are set as (55, 1.2, 0.02), (10, 0.9, 0.01) and (10, 0.9, 0.01), respectively. The initial

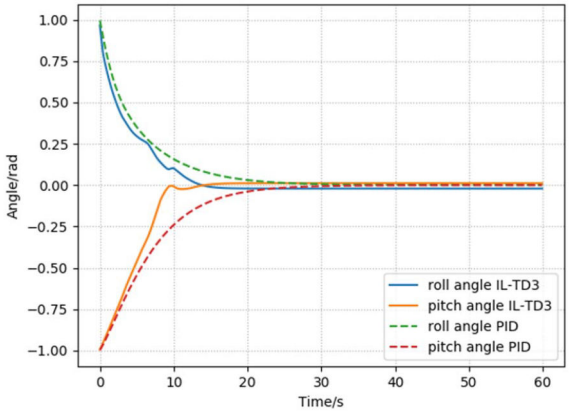

Fig. 8 *Pitch and roll angles by PID and IL-TD3 without current disturbance, initial roll angle is 1.00 radian, initial pitch angle is − 1.00 radian*

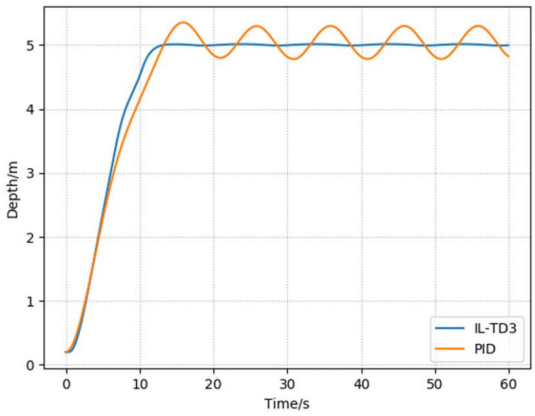

Fig. 9 *Depth by PID and IL-TD3 current disturbance, initial depth is 0.2 m*

state vector of UUV is $\boldsymbol{s}_1 = [0.2 \quad 1.0 \quad -1.0 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$, and the desired state vector is $\boldsymbol{s}_{\exp} = [5.0 \quad 0.0 \quad 0.0 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$. The simulation results of the depth, pitch angle and roll angle given by two different control algorithms are shown in Figs. 7 and 8. In Fig. 7, the initial depth of UUV under PID and IL-TD3 control 0.3 m.

It can be seen that both IL-TD3 and PID can complete the control task under normal circumstances, but the IL-TD3 algorithm only takes 12 s while the PID takes 25 s almost. The depth tracking error in PID control caused by overshoot is 0.3 m, from 12 to 22 s. While the IL-TD3 response faster and has no overshoot in const depth and attitude control.

In order to test the robustness of our algorithm, current disturbance is set as

$$\tau_{\mathrm{Noise}} = \left[30\sin\left(\frac{t}{50\pi}\right) \quad 15\sin\left(\frac{t}{50\pi}\right) \quad 15\sin\left(\frac{t}{50\pi}\right)\right]^{T} \tag{28}$$

The disturbance is added to the torques in the Z, M, N directions. Figs. 9 and 10 show the control results of the IL-TD3 and PID algorithms, respectively. The average tracking error of depth is >0.4 m from 12 to 20 s is in PID control. The average tracking error of depth is proposed algorithm is <0.1 m, from 10 to 60 s in proposed algorithm. The average tracking error of roll and pitch angles are >0.1 rad in PID control. The average tracking error of roll and pitch angles are <0.05 rad in proposed algorithm. Compare with PID, the average tracking error of depth is reduced by 75%, the average tracking error of attitude angles are reduced by 50%. It can be seen that the IL-TD3 algorithm can still perform the control tasks well. However, the PID control algorithm can't complete control task in this case. Fig. 11 shows the outputs of the four thrusters of UUV using IL-TD3 control algorithm and PID control algorithm. It can be seen thatthe thruster output by IL-TD3 is

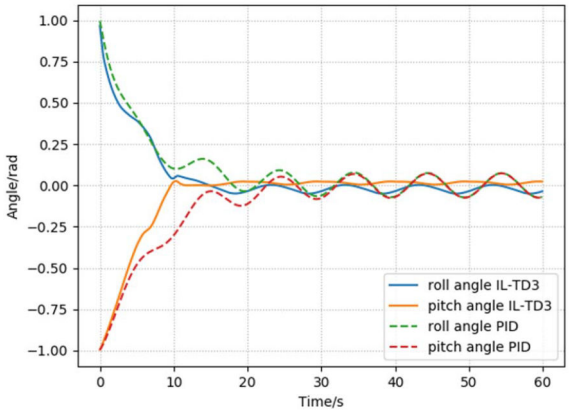

Fig. 10 *Pitch and roll angles by PID and IL-TD3 with current disturbance, initial roll angle is 1.00 rad, initial pitch angle is − 1.00 radian*

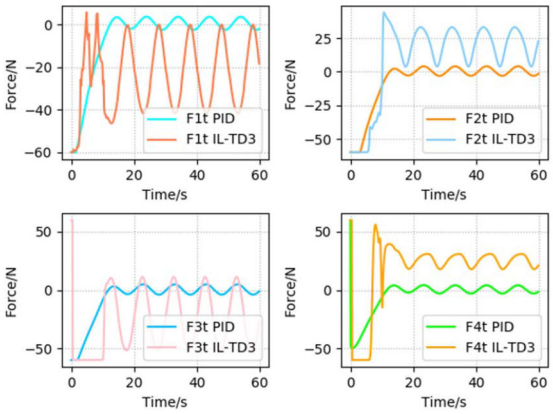

Fig. 11 *Thruster outputs by PID and IL-TD3 control with current disturbance*

different from PID. The four thrusters vary in different amplitude to maintain the control performance with current disturbance.It can be concluded that the robustness of IL-TD3 control is better than that of PID control in this control task. While IL-TD3 algorithm takes more thruster usage than PID algorithm.

In order to test the fault-tolerant ability of the proposed control algorithm in the case of UUV thruster fails, the fourth thruster is assumed to be failed and the thrust is reduced to 50%. The fault simulation method is the same as [23]. In this case, the depth, pitch angle and roll angle of the UUV are shown in Figs. 12 and 13. The thruster outputs are shown in Fig. 14. It can be seen that the IL-TD3 control method has good fault tolerant control capability in the case of thruster fault, the average tracking error of depth is 0.1 m and the average tracking error of attitude angles are about 0.1 rad. The tracking error are not much different from normal.

5.2 Control task 2: depth trajectory tracking control

In this control task, the control law is designed to make the UUV tracking the reference trajectory. The reference depth trajectory ref_t is set as

$$\text{ref}_t = \begin{cases} 2.75\ (\text{m}) & \text{if depth} \geq 2.75\ (\text{m}) \\ 2\sin\left(\dfrac{t}{30}\right) + 2\ (\text{m}) & \text{else} \\ 1.25(\text{m}) & \text{if depth} \leq 1.25\ (\text{m}) \end{cases} \tag{29}$$

During training this control task, the hyperparameters are roughly the same as control task 1, and only the reward function and input state have changed. $\boldsymbol{s}_t = [z_t \quad w_t]$ represents the depth, the velocity in the z-direction at time t. The initial state vector of UUV is $\boldsymbol{s}_1 = [2.0 \quad 0.0]$. The reward function is designed as (see (30)) . The IL-TD3 algorithm, the TD3 algorithm and the DDPG algorithm are

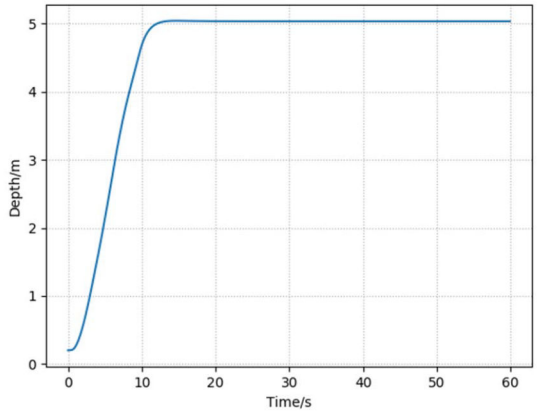

Fig. 12 *Depth by the proposed method with thruster fault, initial depth is 0.2 m*

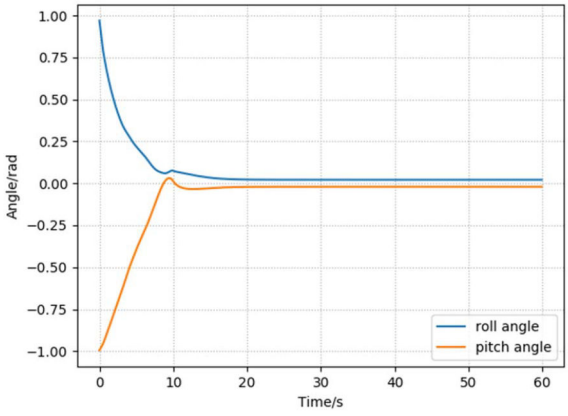

Fig. 13 *Pitch and roll angles by the proposed method with thruster fault, initial roll angle is 1.00 radian, initial pitch angle is − 1.00 radian*

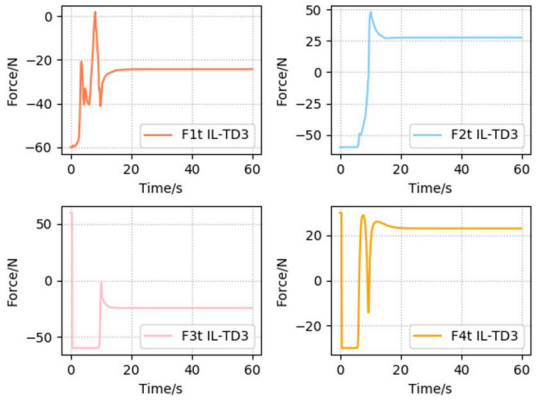

Fig. 14 *Thruster outputs by the proposed method with thruster fault*

employed to train the UUV depth trajectory tracking control task, respectively. The actor network is trained by the supervised learning method with 100 episodes. As shown in Fig. 15, the average loss decreased as the training episodes increased, changed from about 0.44 to about 0.04. In RL, the average reward curves of each algorithm with training 200 episodes are shown in Fig. 16. It can be seen that due to pre-training, IL-TD3 receives higher rewards than TD3 and DDPG at the beginning of training. The least reward of IL-TD3, TD3 and DDPG are about −9, −30 and −36, the biggest reward of three algorithms are about −0.5, −0.6, −2.5. During the training process, after about 100 episodes, the IL-TD3 and TD3 can stabilise about the maximum reward, while DDPG still have drastic changes. The convergence rate of IL-TD3 algorithm during training is about double that of DDPG and TD3. In the whole training process, the IL-TD3 algorithm achieves the best performance, and the DDPG algorithm performs the worst.

$$r_t = \begin{cases} -5|\mathrm{ref}_t - z_t| - 0.1w_t^2 - 0.01\sum_{i=1}^{4}|F_{it}| - 4\times10^{-4}\parallel F_{t-1} - F_t \parallel & \text{if } |\mathrm{ref}_t - z_t| < 1.0 \\ -50 & \text{if } |\mathrm{ref}_t - z_t| \geq 3.5 \\ -5(\mathrm{ref}_t - z_t)^2 - 0.1w_t^2 - 0.01\sum_{i=1}^{4}|F_{it}| - 4\times10^{-4}\parallel F_{t-1} - F_t \parallel & \text{else} \end{cases} \tag{30}$$

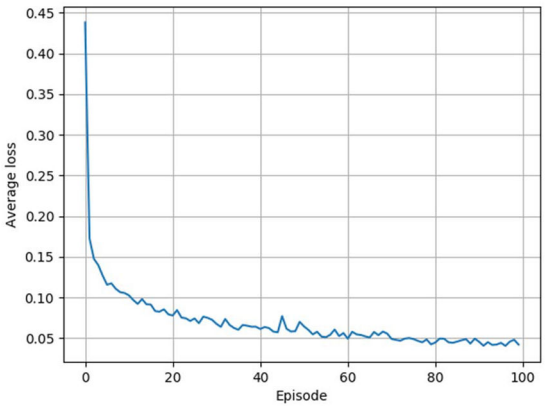

Fig. 15 *Average loss in IL for depth trajectory tracking control*

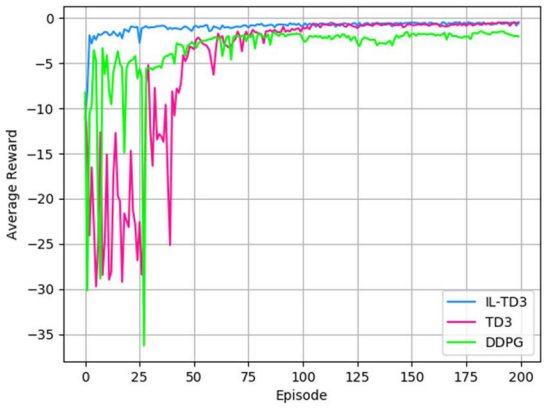

Fig. 16 *Average return reward curves in DRL for depth trajectory tracking control*

Table 4 Sampled rewards of three algorithms in depth trajectory tracking control

Episodes	IL-TD3	TD3	DDPG
1–20	−11.10	−9.30	−8.22
21–40	−1.52	−23.15	−4.89
41–60	−1.19	−8.11	−3.95
61–80	−0.98	−3.47	−2.12
81–100	−0.72	−1.80	−1.64
101–120	−0.83	−0.99	−2.41
121–140	−0.62	−0.94	−2.15
141–160	−0.70	−0.56	−2.53
161–180	−0.59	−0.67	−1.80
181–200	−0.52	−0.60	−1.69

The IL-TD3 algorithm converges faster than TD3 and the learning process is very stable.

Consider the reward curves of three algorithms are hard to intuitive analysis. The average rewards of three algorithms in 200 episodes are sampled per 20 episodes. The sampled rewards are shown in Table 4, it can be concluded that the IL-TD3 is the best algorithm for training this control task among them.

The model trained by IL-TD3 is saved to implement the depth trajectory tracking control task and is compared with the PID algorithm. The PID parameters K_p, K_i, K_d is set as $(90, 1.2, 0.01)$. In

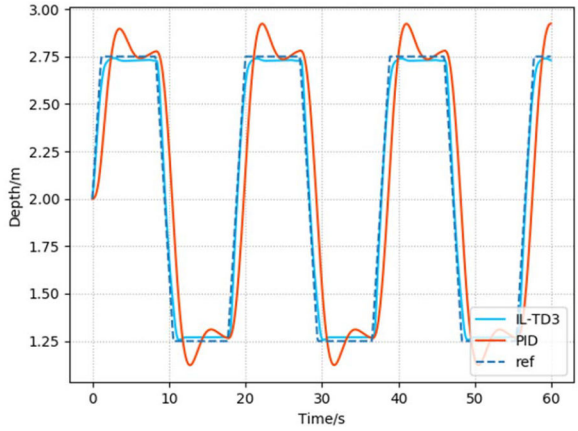

Fig. 17 *Depth trajectory tracking results by IL-TD3 and PID without current disturbance, initial depth is 2.00 m*

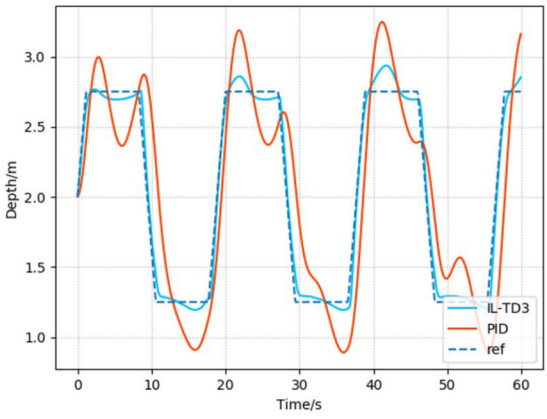

Fig. 18 *Depth trajectory tracking results by IL-TD3 and PID with current disturbance, initial depth is 2.00 m*

the case of no-current disturbance, the depth trajectory control of the IL-TD3 and PID algorithms are shown in Fig. 17. It can be seen that the UUV's trajectory controlled by IL-TD3 algorithm almost identical to the reference trajectory, completes the control task well. However, due to the overshoot of the PID, the overshoot amount is about 0.15 m, the control performance is unsatisfactory.

Then, the current disturbance $\boldsymbol{\tau}_{\mathrm{noise}} = [30\cos(t/50\pi)0\ \ 0]^{\mathrm{T}}$ is added in the vertical direction to test the robustness of IL-TD3 control in depth trajectory tracking control. As shown in Fig. 18, the proposed algorithm can still track most of the reference depth trajectory with current disturbance, with average tracking error of depth is about 0.15 m. While the depth error under PID control algorithm is about 0.5 m. The average tracking error of depth is reduced about 70% in proposed algorithm than in PID. It can conclude that the proposed algorithm can maintain a good robustness in this control task. The thruster outputs of two algorithms are shown in Fig. 19. The usage of thrusters under the proposed algorithm is about 50% more than the usage of thrusters. It outperforms the PID algorithm in this control task.

In the last, the ability of the IL-TD3 algorithm for fault-tolerant control is also tested. It is assumed that the fourth thruster failed and could not work at all at the beginning of the control process. In this case, the depth trajectory tracking control results is shown in Fig. 20. In addition, the thruster outputs are shown in Fig. 21. The average tracking error of depth is about 0.15 m, not much different

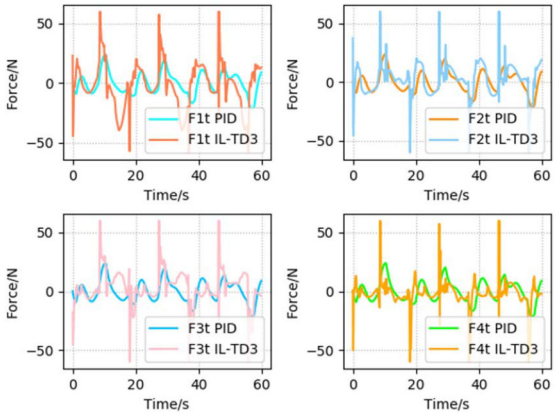

Fig. 19 *Thruster outputs by IL-TD3 with current disturbance*

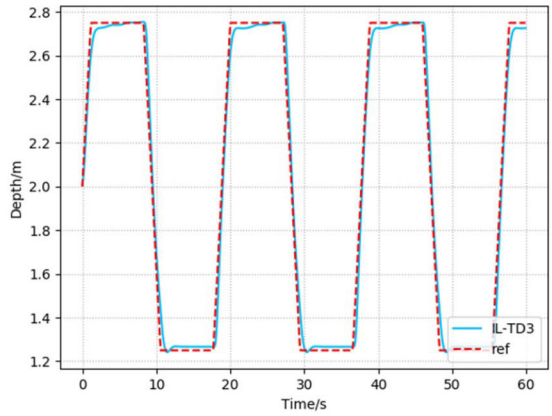

Fig. 20 *Depth trajectory tracking results by the proposed method with thruster fault, initial depth is 2.00 m*

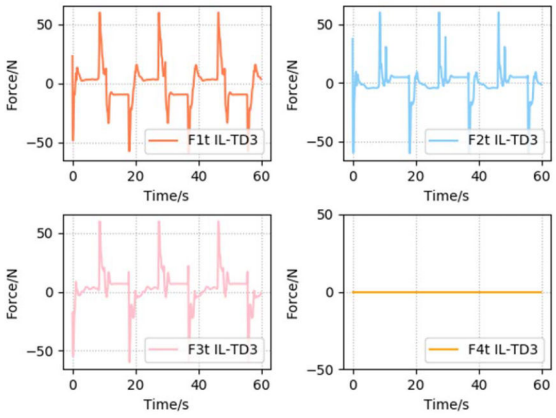

Fig. 21 *Thruster outputs by the proposed control method with thruster fault*

from normal case. It can be seen that the IL-TD3 algorithm can allocate thruster thrust wisely, make the UUV still perform the control task well even when the thruster fault happens.

6 Experiments

To further demonstrate the proposed method, tank experiments based on the underwater vehicle BlueROV2 Heavy are conducted. BlueROV2 Heavy is a ROV developed by Blue Robotics. As shown in Fig. 22, it is composed with four vertical thrusters and four horizontal thrusters like ODIN UUV.

The BlueROV2 Heavy is equipped with some sensors to obtain state information, the state information are processed on a PixHawk autopilot by the open source software ArduSub, then the information are uploaded to principal computer by Raspberry Pi on-board computer, Raspberry Pi on-board computer use MAVLink protocol to complete communication task. The control

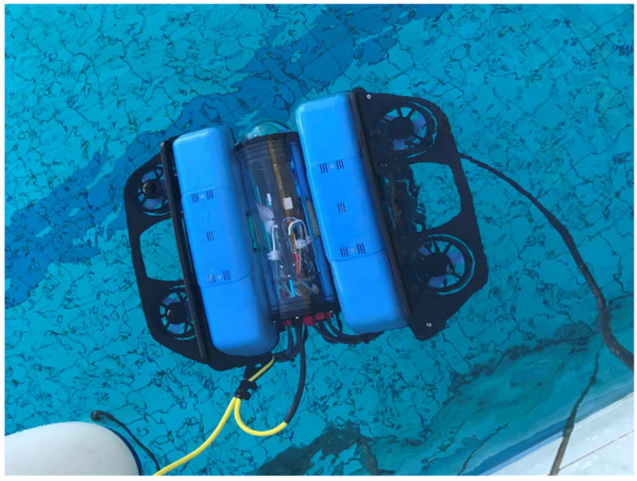

Fig. 22 *Tank test of BlueROV2 Heavy*

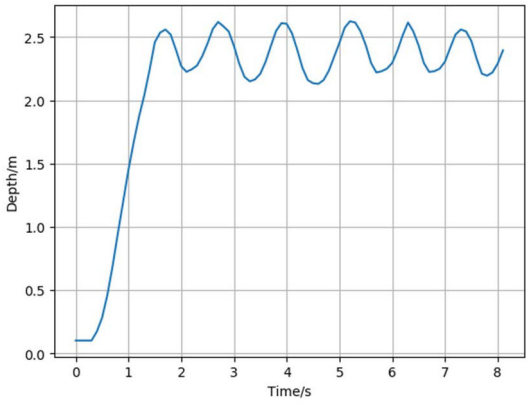

Fig. 23 *Depth of BlueROV2 during the tank test*

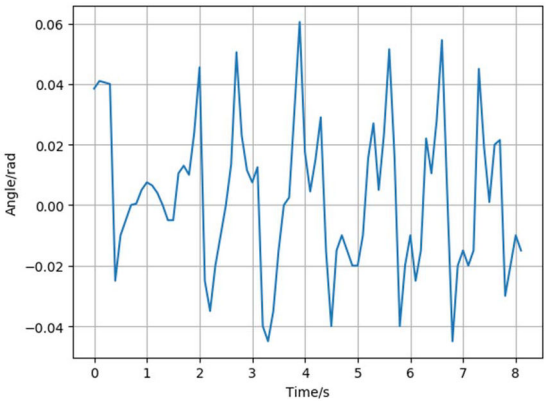

Fig. 24 *Pitch angle of BlueROV2 during the tank test*

command is issued by principal computer, converted to pulse-width modulation (PWM) value to the ROV's thrusters.

Constant depth and attitude control task are completed in the tank test. During the experiments, the ROV's software are modified to achieve individual drive of ROV's thrusters. The IL-TD3 algorithm runs on principal computer, the control strategy is same as control task 1 in simulation part. Thrusters forces are computed on principal computer, then PixHawk delivers PWM signals to each thruster, respectively.

After training, the experimental data are shown in Figs. 23–25. The depth of ROV is quickly achieved expected value, take about 2 s. The data shows that the vehicle has basic depth and attitude control ability, while it still has large oscillation amplitude. The depth error is about 0.25 m, the pitch angle error is about 0.05 rad and the roll angle error is about 0.02 rad.

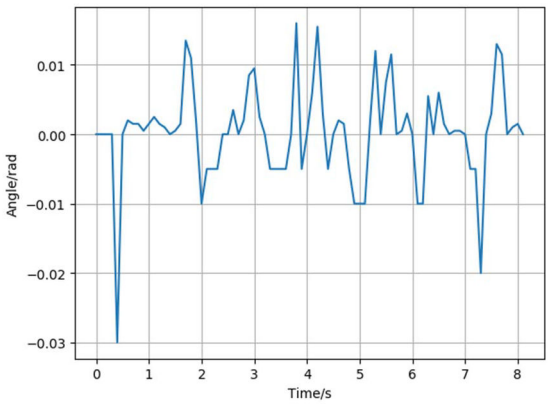

Fig. 25 *Roll angle of BlueROV2 during the tank test*

7 Conclusion

In this paper, the IL-TD3 algorithm for UUVs motion control is presented. This algorithm combines the advantages of IL and DRL. Using the data of PID algorithm as expert data to pre-train the actor network, it can accelerate the training process of DRL and the control performance of trained model is better than PID. From the simulations of two control tasks, it can be concluded that IL-TD3 algorithm has good robustness compared with PID, and achieved good performances in fault-tolerant control. However, the IL-TD3 algorithm still has some shortcomings. In the IL part, behaviour cloning based on supervised learning alone is not enough to make actor network achieve good effects in pre-training and the usage of thrusters in IL-TD3 are more than these in PID. In the future, more effective IL algorithms will be adapted. IL-TD3 algorithm will be compared with some popular DRL algorithm, such as proximal policy optimisation and soft actor–critic.

8 Acknowledgments

This project was supported by the National Natural Science Foundation of China under grant number 51839004 and International Academic Cooperation and Exchange Project of Shanghai under grant number 18550720100 and Capacity Building Project of Shanghai Local Colleges and Universities under grant no. 19040501600.

9 References

[1] Li, X., Zhu, D.: 'An adaptive SOM neural network method for distributed formation control of a group of AUVs', *IEEE Trans. Ind. Electron.*, 2018, **65**, (10), pp. 8260–8270

[2] Wang, Z., Qin, Y., Hu, C., *et al.*: 'Fuzzy observer-based prescribed performance control of vehicle roll behavior via controllable damper', *IEEE Access*, 2019, **7**, pp. 19471–19487

[3] Fan, S., Li, B., Xu, W., *et al.*: 'Impact of current disturbances on AUV docking: model-based motion prediction and countering approaches', *Ocean. Eng.*, 2018, **43**, (4), pp. 888–904

[4] Zhao, W., Qin, X., Wang, C.: 'Yaw and lateral stability control for four-wheel steer-by-wire system', *IEEE/ASME Trans. Mechatronics*, 2018, **23**, (6), pp. 2628–2637

[5] Shen, C., Shi, Y., Buckham, B.: 'Path-following control of an AUV: A multi objective model predictive control approach', *IEEE Trans. Control Syst. Technol.*, 2019, **27**, (3), pp. 1334–1342

[6] Shen, C., Shi, Y., Buckham, B.: 'Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control', *IEEE Trans. Ind. Electron.*, 2018, **65**, (7), pp. 5796–5805

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

[8] Lillicrap, T.P., Hunt, J.J., Pritzel, A., *et al.*: 'Continuous control with deep reinforcement learning'. Int. Conf. on Learning Representations (ICLR), San Juan, Puerto Rico, May 2016

[9] Silver, D., Liver, G., Heess, N., *et al.*: 'Deterministic policy gradient algorithms'. Int. Conf. on Machine Learning (ICML), Beijing, People's Republic of China, November 2014

[10] Fujimoto, S., van Hoof, H., Meger, D.: 'Addressing function approximation error in actor-critic methods'. Int. Conf. on Machine Learning (ICML), Stockholm, Sweden, July 2018

[11] Todorov, E., Erez, T., Tassa, Y.M.: 'A physics engine for model-based control'. Intelligent Robots and Systems (IROS), Vilamoura, Portugal, October 2012, pp. 5026–5033

[12] Liu, Y., Gupta, A., Abbeel, P., *et al.*: 'Imitation from observation: learning to imitate behaviors from raw video via context translation'. IEEE Int. Conf. on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 2018, pp. 1118–1125

[13] Yi, M., Xu, X., Zeng, Y., *et al.*: 'Deep imitation reinforcement learning with expert demonstration data', *J. Eng.*, 2018, **2018**, (16), pp. 1567–1573

[14] Cui, R., Yang, C., Li, Y., *et al.*: 'Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning', *IEEE Trans. Syst., Man, Cybern.: Syst.*, 2017, **47**, (6), pp. 1019–1029

[15] Carlucho, I., Paula, M.D., Wang, S., *et al.*: 'Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning', *Robot. Auton. Syst.*, 2018, **107**, pp. 71–86

[16] Wu, H., Song, S., You, K., *et al.*: 'Depth control of model-free AUVs via reinforcement learning'. *IEEE Trans. Syst., Man, Cybern.: Syst.*, 2019, **49**, (12), pp. 2499–2510

[17] Qiao, L., Yi, B., Wu, D., *et al.*: 'Design of three exponentially convergent robust controllers for the trajectory tracking of autonomous underwater vehicles', *Ocean Eng.*, 2017, **134**, pp. 157–172

[18] Sutton, R.S., Barto, A.G.: '*Reinforcement learning: an Introduction*' (MIT Press, Cambridge, MA, USA, 1998), pp. 6–12

[19] Vaandrager, M., Grondman, I., Busoniu, L., *et al.*: 'Efficient model learning methods for actor–critic control', *IEEE Trans. Syst., Man, Cybern.*, 2012, **42**, (3), pp. 591–602

[20] Shi, W., Song, S., Wu, C., *et al.*: 'Multi Pseudo Q-learning-based deterministic policy gradient for tracking control of autonomous underwater vehicles', *IEEE Trans. Neural Netw. Learn. Syst.*, 2019, **30**, (12), pp. 3534–3546

[21] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization'. Int. Conf. on Learning Representations (ICLR), San Diego, CA, USA, May 2015

[22] Podder, T.K., Sarkar, N.: 'Fault-tolerant control of an autonomous underwater vehicle under thruster redundancy', *Robot. Auton. Syst.*, 2001, **34**, (1), pp. 39–52

[23] Chu, Z., Meng, F., Zhu, D., *et al.*: 'Fault reconstruction using a terminal sliding mode observer for a class of second-order MIMO uncertain nonlinear systems', *ISA Trans.*, 2020, **97**, pp. 67–75, doi: 10.1016/j.isatra.2019.07.024