Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

Deep Reinforcement Learning Based Optimal Trajectory Tracking Control of Autonomous Underwater Vehicle

Runsheng Yu*, Zhenyu Shi*, Chaoxing Huang*, Tenglong Li*, Qiongxiong Ma
*These authors contributed equally
South China Normal University, Guangzhou 510631, P. R. China
E-mail: 20143201002@m.scnu.edu.cn, 20143201094@m.scnu.edu.cn, goldenhwong@gmail.com, 20143201060@m.scnu.edu.cn, robotteam@qq.com

Abstract: The aim of this paper is to solve the control problem of trajectory tracking of Autonomous Underwater Vehicles (AUVs) through using and improving deep reinforcement learning (DRL). The deep reinforcement learning of an underwater motion control system is composed of two neural networks: one network selects action and the other evaluates whether the selected action is accurate, and they modify themselves through a deep deterministic policy gradient(DDPG). These two neural networks are made up of multiple fully connected layers. Based on theories and simulations, this algorithm is more accurate than traditional PID control in solving the trajectory tracking of AUV in complex curves to a certain precision.

Key Words: Autonomous Underwater Vehicles, Optimal Control System, Deep Reinforcement Learning

1 Introduction

In recent years, AUVs have been widely utilised in ocean exploration and the protection of the marine environment, and their status is increasingly important[1][24][25]. Some extremely dangerous tasks are made safe by the accurate control of AUV, such as exploring for submarine oil, repairing submarine pipelines, as well as tracing and recording the positions of torpedoes. However, because of the complexity of underwater environments, the autonomous control of AUVs is nonlinear, as their motions are easily influenced by flow and hydraulic resistance, making accurate control difficult. By referring to classical control theories, many specialists and scholars have studied underwater nonlinear systems and achieved variable results. For example, Min J. Kim et al. put forward an underwater hovering and tracking control method based on the fuzzy PID with an accuracy of 1 meter[2]. Lu Wang et al. proposed an underwater nonlinear tracking control method based on sliding-mode control[3]. AUV motion control has developed from simple feedback control to the nonlinear stage of advanced robust, intelligent control[4].

With the rapid development of artificial intelligence in recent years, machine learning is widely used in different fields [16], especially in the control field. Many scholars are starting to pay attention to the application of AI in AUV control. Mariano De Paula and Gerardo G. Acosta have proposed a self-adaptive reinforcement learning tracking control algorithm of AUVs [5]. However, this algorithm is too general for use in AUV control. A. El-Fakdi et al. came up with an online policy learning control method of AUVs based on reinforcement learning of stochastic gradient descent[6]. One of the primary goals of AI is to solve complex tasks from unprocessed, high-dimensional and sensory input [8]. The control of AUVs is also going in this direction.

To improve the performance of robot control problems when carrying out untreated complex tasks with high dimensionality and perceptible input, David Silver et al. proposed a deterministic policy gradient (DPG) algorithm. The experiment results demonstrate a significant advantage of using a deterministic policy gradient over a stochastic policy gradient[7]. DPG can also be used to solve other nonlinear optimization problems where the method of stochastic gradient descent is ineffective. In 2015, Mnih et al. proposed a reinforcement learning algorithm called deep Q network(DQN) which completes the task of playing Atari computer games perfectly [8]. However, DQN cannot be directly applied to continuous problems when the control of high dimensional motions is involved. Therefore, Timothy P. Lillicrap et al. designed an algorithm based on DPG called DDPG (deep deterministic policy gradient) in ICLR in 2016[8]. Inheriting DPG, this algorithm uses for reference the actor-critic method in the DQN algorithm, and has high stability and robustness in solving multiple challenging physical control problems [17].

This paper aims to produce a method with better performance to control AUVs and to explore the application of the DRL algorithm in underwater control. We put forward the underwater tracking control algorithm of AUVs and create the autonomous underwater vehicle control model by using the DRL algorithm. We also carry out a simulation verification of the control effect.

The rest of this paper will be structured as follows: The second part introduces the underwater dynamic equation of AUVs and gives a brief description of the system. The third part explains the structure of neural networks as well as the training process of the control neural network(actor) and the evaluation neural network(critic). The fourth part shows the analysis of the stability of the system. The fifth part describes the training and simulation. Finally, we draw conclusions.

This work is supported by Natural Science Foundation of Guangdong Province, China under Grant 2016A030313456, Science and Technology Planning Project of Guangdong Province, China under Grant 2015B090920003, 2016B090917002, 2016B090926004, South China Normal University National Training Programs of Innovation and Entrepreneurship under Grant 201610574038, Youth Teacher Science Cultivation Foundation of South China Normal University under Grant 15KJ13.

2 The Dynamic Model of Autonomous Underwater Vehicles

This section presents the general underwater dynamic equation of AUVs. The control system of AUVs will also be introduced in this section.

2.1 Nonlinear dynamic equation and control system

In this paper, we consider the AUV moves in a horizontal plane.

The starting point is defined as the origin and the earthxed inertial frame $\{Ox^by^bz^b\}$ is set according to a left-handed system.

The underwater dynamic equation of AUVs[9][13] can be expressed as follows :

$$M\dot{v} + C(v)v + D(v)v + g(\eta) + \delta = \tau \tag{1}$$

$$\Re(\varphi)v = \dot{\eta} \tag{2}$$

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}$$

$$\Re(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C(v) = \begin{bmatrix} 0 & 0 & -m_{22}v \\ 0 & 0 & m_{11}u \\ m_{22}v & -m_{11}u & 0 \end{bmatrix}$$

$$D(v) = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix}$$

$$g(\eta) = [g_x(\eta), g_y(\eta), g_z(\eta)]^T$$

Among which, M is the mass matrix, $C(v)$ is the centrifugal and Coriolis matrix, $D(v)$ is the damping matrix, $g(\eta)$ is the gravity and buoyancy matrix, and τ is the target input which refers to the force of motor. δ is the model uncertainty vector, which is induced by disturbances.

The position and velocity can be expressed as follows:

$$\eta = [x, y, \varphi]^T$$

$$v = [u, v_1, r]^T$$

Where u is the velocity in surge, v_1 is the velocity in sway, and r is the velocity in yaw, while x and y are the linear position and φ is the direction of AUVs in an earthxed inertial frame.

The AUV movement in the horizontal plane is shown in figure 1.

Assumption I: The parameters of the gravity and buoyancy matrix $[g_x(\eta), g_y(\eta), g_z(\eta)]$ are constants.

This paper discusses the horizontal motions of AUVs at a constant depth and it is apparent that the above parameters are constants.

When the mass matrix is not a singular matrix,

$$\dot{v} = M^{-1}(\tau - D(v)v - g(\eta) - C(v)v - \delta) \tag{3}$$

$$\Re(\varphi)v = \dot{\eta} \tag{4}$$

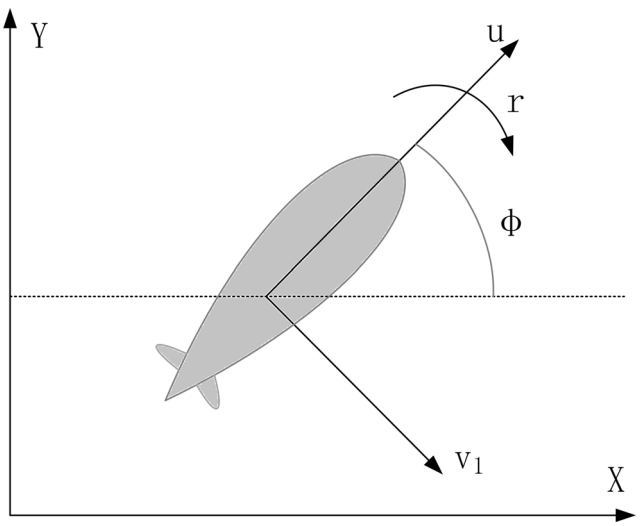

Fig. 1: AUV movement on the horizontal plane.

By applying the Taylor expansion of the first order, there are

$$v(t+1) = M^{-1}\tau + M^{-1}G(t) \tag{5}$$

$$\eta(t+1) = \Re(\varphi(t))v(t) \tag{6}$$

Where

$$G(t) = (-D(v(t))v - g(\eta(t)) - C(v(t))v(t) - w)$$

t is a certain moment of the system. The controller can be set as follows:

$$\tau(t) = \mu(v(t), \eta(t)) \tag{7}$$

The force of AUVs is determined by its velocity and position of previous moment, which means that the controller $\mu(\cdot)$ is a map from state to action. To simplify it, let

$$s_t = [v(t), \eta(t)]^T$$

In figure 2, the agent that adapts the policy is constituted by the networks of actor and critic. The actor network is trained to work with the control policy (force) as controller. In addition, the critic network is trained and the output is used as a parameter of actor network to evaluate the training performance of the controller. Noise from the environment is taken into account as disturbance during the entire trial.

3 The Proposing and Evaluating of Actor and Critic Neural Networks

In this section, we introduce the multiple fully connected layers neural network, the control policy function μ and the critic function Q. Then, two neural networks are given to replace the control policy and critic function respectively. The control algorithm is also given below.

3.1 Multiple fully connected layers[26]

The fully connected layer is presented as

$$y = \sigma(w^T x + b) + \varepsilon \tag{8}$$

Where w^T is the weight and b is the bias. x is defined as the input of the layer while y is the output. The function of

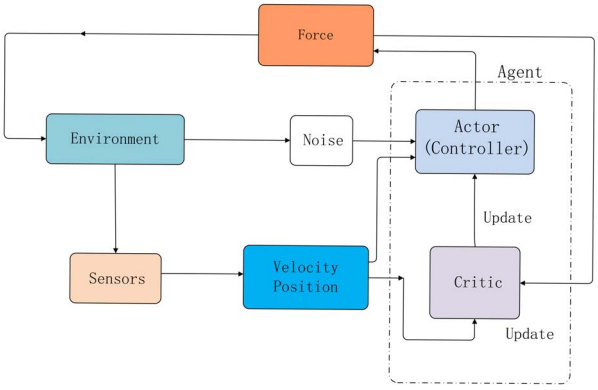

Fig. 2: Overview of the AUVs control system

Relu[23] is chosen to be the activate function as σ, and ε is the error function.

The whole network is constituted by a multiple fully connected network and it can be known from [11] that

$$\|\varepsilon\| \leq b_1 \, , b_1 \in Const$$

3.2 Control policy and the optimal control policy

It has been mentioned in (7)

$$\mu(s_t) = \tau \tag{9}$$

τ also can be defined as a_t(action)

As for the underwater trajectory tracking control, the target will make the system move along the desired trajectory state $\hat{s}_t$ by using the control policy at any moment.

It means the actor function is designed to stabilize the tracking error in an optimal manner by minimizing the reward function:

$$r(s_t, a_t) = [-I(\hat{s}_t - s_t)^2 - \Lambda a_t^2] \tag{10}$$

I and Λ are all positive-definite functions.

To take the influence of possible future into consideration, the long-term reward function is defined as:

$$R(s_t, a_t) = \int_t^{\infty} \gamma^{-(\kappa - t)} r(s_t, a_t) d\kappa \tag{11}$$

γ is the discount factor$(0 < \gamma < 1)$, which is used to weaken the influence of possible future state[10].

Therefore, this problem can be described as the following mathematical problem.

Problem 1: **find** $\underset{s_t, a_t}{\operatorname{argmin}} \{R(s_t, a_t)\}$ **with the constraint conditions**:

$$s.t. \quad a_{\min} \leq a_t \leq a_{\max}$$
$$s_{\min} \leq s_t \leq s_{\max}$$

3.3 Critic function[22]

In order to solve Problem 1, critic function is defined as:

$$Q(s_t, a_t) = R(s_t, a_t) = \int_t^{\infty} \gamma^{-(\kappa - t)} r(s_t, a_t) d\kappa \tag{12}$$

Discretize (12)

$$R(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{(i-t)} r(s_t, a_t) \tag{13}$$

It can be known from the Bellman equation [21]

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) \tag{14}$$

The optimal critic function is

$$\hat{Q}(s_t, a_t) = \underset{s_t, a_t}{\operatorname{argmax}} \left(Q(s_t, a_t)\right) \tag{15}$$

Replace $Q(s_t, a_t)$ by using neural network

$$Q(s_t, a_t|\omega) \tag{16}$$

In order to evaluate a policy, we must firstly get the optimal critic function $\hat{Q}(s_t, a_t)$. We define the $Loss$ function as

$$Loss = \frac{1}{2}(y_t - Q(s_t, a_t|\omega))^2 \tag{17}$$

$$y_t = r(s_t, a_t) + \gamma Q(s_t, a_t|\omega) \tag{18}$$

The optimal critic function $\hat{Q}(s_t, a_t)$ can be found by minimizing the value of $Loss$ function.

Thus, the policy gradient algorithm [7] is applied here by sampling a batch of (s_t, a_t) and computing the average $Loss$ function.

$$\bar{Loss} = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - Q(s_i, a_i|\omega)\right)^2 \tag{19}$$

Find gradient

$$\nabla_\omega \bar{Loss} = -\frac{2}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i|\omega)) \frac{\partial Q(s_i, a_i|\omega)}{\partial \omega} \tag{20}$$

Update weight by gradient descent methods

$$\omega_{t+1} = \omega_t + \alpha \cdot \nabla_\omega Loss \tag{21}$$

Where α is the learning rate.

3.4 Actor function[22]

When we find the critic function, we can use it to update the actor function.

Replace $\mu(s_t)$ by using neural network $\mu(s_t|\theta)$ and substitute $a_t = \mu(s_t|\theta)$ into $Q(s_t, a_t|\omega)$. It has

$$J = Q(s_t, \mu(s_t|\theta)|\omega) \tag{22}$$

Differentiate (22)

$$\nabla_\theta J = \frac{\partial Q(s_t, \mu(s_t|\theta)|\omega)}{\partial a_t} \cdot \frac{\partial \mu(s_t|\theta)}{\partial \theta} \tag{23}$$

Use ADAM [12] to update the network:

$$m_{t+1} = \wp \cdot m_t + (1 - \wp) \nabla_\theta \mu$$

$$\Im_{t+1} = \beta \cdot \Im_t + (1 - \beta) \nabla_\theta \mu$$

$$\hat{m}_t = \frac{m_t}{1 - \wp^t}$$

$$\hat{\Im}_t = \frac{\Im_t}{1 - \beta^t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{1}{\sqrt{\hat{\Im}_t}} \hat{m}_t$$

Where $\wp$ and β are the learning rate.

To make both the actor and critic neural networks update their weights in a more "soft" way. We create a copy of the actor and critic networks, named $Q'(s_t, a_t|\omega')$ and $\mu'(s_t|\theta')$[8]. The whole algorithm is as Algorithm 1. The algorithm will only stop when the $\sigma < \varepsilon_r$, where σ is the standard deviation of total reward function in each 100 episodes and ε_r is the threshold.

Algorithm 1: DRL algorithm

Initialize the networks of $Q(s_t, a_t|\omega)$ and $\mu(s_t|\theta)$
with weights ω and θ.
Initialize the copy networks of $Q'(s_t, a_t|\omega')$ and $\mu'(s_t|\theta')$
with weights ω' and θ'.
Initialize the replay buffer R
while ($\sqrt{\frac{\sum_{j=M-100}^{M}\sum_{i=1}^{T}[r_j(s_i,a_i)-\bar{r}(s_t,a_t)]^2}{100T}} > \varepsilon_r$) **do**
(M is the current training episode)
  Initialize the state s_0
  for t=1, T **do**
    Choose actor $a_t = \mu(s_t|\theta)$
    Get the state s_{t+1} according to the environment
    compute $r(s_t, a_t)$
    store in transition $R(s_t, a_t, r_t, s_{t+1})$
    Randomly select N arrays from R
    compute $y_i = r_i + \gamma Q'(s_i, a_i|\omega')$
    compute $Loss = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\omega))^2$
    compute $\nabla_\omega Loss = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\omega))\frac{\partial Q(s_i,a_i|\omega)}{\partial \omega}$
    update weight $\omega_{t+1} = \omega_t + \alpha \cdot \nabla_\omega Loss$
    compute $\nabla_\theta J = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial Q(s_i,a_i|\mu(s_i|\theta))}{\partial a_i} \cdot \frac{\partial \mu(s_i|\theta)}{\partial \theta}$
    compute $m_t = \wp \cdot m_{t-1} + (1-\wp)\nabla_\theta \mu$
    compute $\Im_t = \beta \cdot \Im_{t-1} + (1-\beta)\nabla_\theta \mu$
    compute $\hat{m}_t = \frac{m_t}{1-\wp^t}$
    compute $\hat{\Im}_t = \frac{\Im_t}{1-\beta^t}$
    update weight $\theta_{t+1} = \theta_t - \eta\frac{1}{\sqrt{\hat{\Im}_t}}\hat{m}_t$
    update weight $\omega' = \rho\omega + (1-\rho)\omega'$
    update weight $\theta' = \rho\theta + (1-\rho)\theta'$
    (ρ is learning rate)
  end for
end while
end

4 Analysis of the Stability of the Control System

In this section, the analysis of the stability of the control system will be given.

Before analysing the stability, we introduce the following assumptions.

Assumption II: The damping matrix $D(v)$ of AUVs is a positive definite matrix[14][20].

Assumption III: The velocity, acceleration and the derivative of acceleration should be significantly smaller than the changing rates of weight, which is

$$\left|\frac{\partial s_t}{\partial t}\right| \ll \left|\frac{\partial \theta}{\partial t}\right| \tag{24}$$

The running speed of CPU should be significantly faster than the change rates of other external environments[15].

Assumption IV: There should be the following numerical relationships:

When $s_t - \hat{s}_t \geq 0$, $a_t \leq 0$
When $s_t - \hat{s}_t < 0$, $a_t > 0$

It can easily be found that the AUV will travel backward when the boat is moving away from the target position, and it will travel forward when the boat is approaching the target position.

The analysis of the stability is as follows:

Combine (1)(7)(9)

$$\mu(s_t|\theta) = D(s_t)s_t + m\dot{s}_t + g(\eta) + C(s_t)s_t \quad (25-1)$$

It is easily known that $D(s_t)$ is not a singular matrix.

Applying transposition on $(25-1)$, we can get:

$$s_t = \frac{\mu(s_t|\theta) - m\dot{s}_t - C(s_t)s_t - g(\eta)}{D(s_t)} \quad (25-2)$$

According to assumption I:

$$g(\eta) = Const$$

Take the derivatives of t of $(25-2)$ two sides, we get:

$$\frac{ds_t}{dt} = \frac{D(s_t)(K_2 - \frac{\partial D}{\partial s_t}\frac{\partial s_t}{\partial t}K_1)}{[D(s_t)]^2} \tag{26}$$

Among which,

$$K_1 = \mu(s_t|\theta) - m\dot{s}_t - C(s_t)s_t \tag{27}$$

$$K_2 = \frac{\partial \mu(s_t|\theta)}{\partial s_t}\frac{\partial s_t}{\partial t} + \frac{\partial \mu(s_t|\theta)}{\partial \theta}\frac{\partial \theta}{\partial t} - m\frac{\partial \dot{s}_t}{\partial t} - \frac{\partial C}{\partial s}\frac{\partial s_t}{\partial t} - \frac{\partial s_t}{\partial t} \tag{28}$$

From assumption III, we can know that $\left|\frac{\partial s_t}{\partial t}\right|$ can be ignored. After simplifying the above equations, we can get

$$\frac{ds_t}{dt} = \frac{\frac{\partial \mu(s_t|\theta)}{\partial \theta}\frac{\partial \theta}{\partial t}}{D(s_t)} \tag{29}$$

Now we apply the Lyapunov function,

$$L(t) = \frac{1}{2}(s_t - \hat{s}_t)^2 \tag{30}$$

Among which, $\hat{s}_t$ means the desired value.

Apply derivation of t on Lyapunov function

$$\frac{dL}{dt} = (s_t - \hat{s}_t)\frac{ds_t}{dt} \tag{31}$$

Substitute equation (29) into equation (31)

$$\frac{dL}{dt} = (s_t - \hat{s}_t)\frac{\frac{\partial \mu(s_t|\theta)}{\partial \theta}\frac{\partial \theta}{\partial t}}{D(s_t)} \tag{32}$$

Substitute equation (23) into equation (32)

$$\frac{dL}{dt} = \frac{(s_t - \hat{s}_t)\alpha(\frac{\partial \mu(s_t|\theta)}{\partial \theta})^2\nabla_a Q(s_t, \mu(s_t|\theta)|\omega)}{D(s_t)} \tag{33}$$

The $Loss$ function is achieved by (19):

$$\varepsilon = (r(s_t, a_t) + (\gamma - 1)Q(s_t, a_t|\omega))^2 \tag{34}$$

Apply derivation on both sides of (34)

$$\frac{\partial \varepsilon}{\partial a_t} = 2[r(s_t, a_t) + (\gamma - 1)Q(s_t, a_t|\omega)] \cdot H \tag{35}$$

Where

$$H = [\frac{\partial r(s_t, a_t)}{\partial a_t} + (\gamma - 1)\frac{\partial Q(s_t, a_t|\omega)}{\partial a_t}]$$

If the $Loss$ function has the minimum value, then the result of equation (35) is zero and the optimal solution can be found [18]. Let equation (35) be zero

$$\frac{\partial Q(s_t, a_t|\omega)}{\partial a_t} = \frac{\partial r(s_t, a_t)}{\partial a_t}/(1 - \gamma) \tag{36}$$

Substitute equation (36) into (33)

$$L'(t) = (s_t - \hat{s}_t)\alpha(\frac{\partial \mu(s_t|\theta)}{\partial \theta})^2 \frac{\partial r(s_t, a_t)}{\partial a_t}/(1 - \gamma)D(s_t) \tag{37}$$

From assumption II, we can know $D(s_t) > 0$.

According to the Lyapunov stability principle, take the proper reward function $r(s_t, a_t)$ which satisfies

$$\alpha(\frac{\partial \mu(s_t|\theta)}{\partial \theta})^2 \frac{\partial r(s_t, a_t)}{\partial a_t}/(1 - \gamma)D(s_t) = L'(t) < 0 \tag{38}$$

then the system is stable.

It can be known from (10) that

$$r(s_t, a_t) = [-I(\hat{s}_t - s_t)^2 - \Lambda a_t^2] \tag{39}$$

Substitute equation (39) into equation (38)

$$L'(t) = 2(s_t - \hat{s}_t)\alpha(\frac{\partial \mu(s_t|\theta)}{\partial \theta})^2 a_t \Lambda/(1 - \gamma) \tag{40}$$

$$L'(t) = 2(s_t - \hat{s}_t)(\frac{\partial \mu(s_t|\theta)}{\partial \theta})^2 a_t \Gamma \tag{41}$$

Where

$$\Gamma = \alpha D(s_t)\Lambda/(1\text{-}\gamma) \tag{42}$$

With assumption II, $D(s_t) > 0$. In addtion, $0 < \gamma < 1$, $1 - \gamma > 0$, $\alpha > 0$, $\Lambda > 0$. Then $\Gamma > 0$ is permanently established.

According to assumption IV

When $s_t - \hat{s}_t \geq 0$, $a_t \leq 0$.

When $s_t - \hat{s}_t < 0$, $a_t > 0$.

It means $L'(t) \leq 0$, the system is stable.

5 Simulation Studies

In this section, the results of the simulation verify the validity of underwater optimal path control by using DRL. All the parameters of the AUV model are given in the Appendix. Simulations are performed under a Tensorflow/Ubuntu environment.

The parameters of DRL are set as follows: both the actor and critic neural networks have 3 hidden layers. All the layers are fully connected. Each layer has 100 neurons and 100 bias. The activate function between the hidden layers is Relu, and that between the hidden layer and the output layer is tanh. Dropout layers [19] are used to prevent overfitting.

The learning rates of actor and critic networks are set at 0.001 which is proper for the control systems of AUVs. To quicken the computing rate of the computer, the batch size is set at 128.

The discount factor of reward in (11) is set at 0.99, and the training time is set at 300 steps for each training episode. Over-abundance of steps may greatly decrease the training speed or even stop the learning, while too few steps may cause insufficient sampling and result in a low successful learning rate.

The ramdom disturbtance is $\delta_x \sim N(0, 1.6)$, $\delta_y \sim N(0, 1.6)$, where $N(\psi, \sigma^2)$ is the Gaussian distribution.

Meanwhile, to verify DRL's accuracy and efficiency in controlling the AUV, we compare it with the traditional PID control in the simulation. Comparative simulation on AUV control based on traditional PID is also introduced.

To better evaluate the control system, some error indicators were used and are defined here.

The deviation of actual track and the ideal track on the x axis is defined as:

$$e_x = x - x_d$$

x is the actual track, and x_d is the ideal track.

The deviation of actual track and the ideal track on the y axis is defined as:

$$e_y = y - y_d$$

y is the actual track, and y_d is the ideal track.

The total error is defined as:

$$e_t = \sqrt{e_x^2 + e_y^2}$$

For comparison, we select the PID controller, which can be expressed as

$$U(s) = K_p e(t) + K_d \frac{d}{dt} e(t) + K_I \int_0^t e(\tau) d\tau$$

5.1 The simulation results of the application of DRL in straight-line trajectory tracking in a horizontal plane

The desired straight-line trajectory is as follow:

$$y_d(t) = x_d(t)$$
$$x_d(t) = t$$

Where t is the steps.

The parameters of the PID controller are as follows: $K_p = 1.56$, $K_I = 6.4$, $K_d = 0.002$. These parameters are set based on experience. We set $\eta = [0, 0, 0]^T$ $v = [0, 0, 0]^T$ at the beginning of the simulation.

The simulation results are as follows:

Figure 3 is the trajectory of the DRL(training episode 1000) and the PID controller, and figure 4 is the trajectory of the DRL in different training episodes.

Figure 5 shows the total error of DRL(training episode 1000) and PID controller. Figure 6 shows the total error of DRL in different traning episodes. It can be seen in the figures that more episodes results in better accuracy in the practical trajectory tracking. The performance of the DRL (episode 1000) is more stable and robust than that of the PID controller.

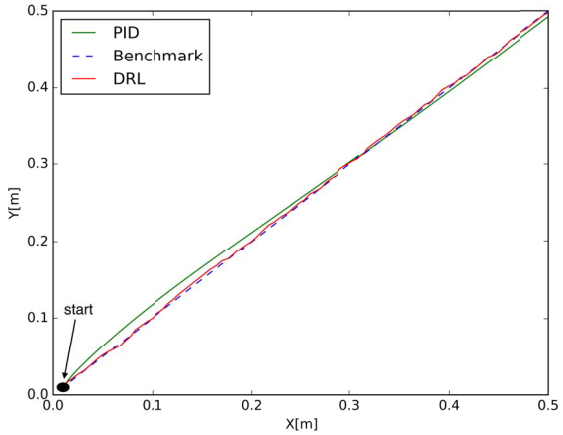

Fig. 3: The trajectory of DRL(training epidode 1000) and PID controller

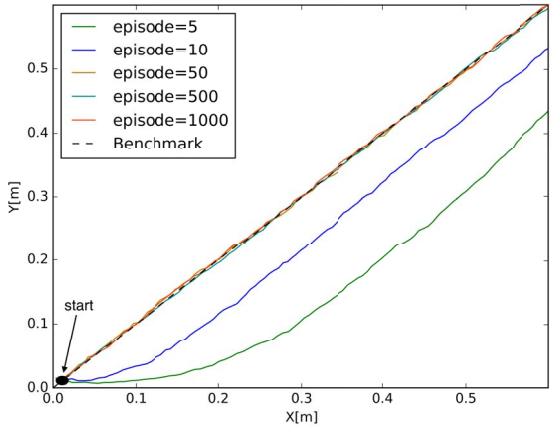

Fig. 4: The trajectory of DRL in different training episodes

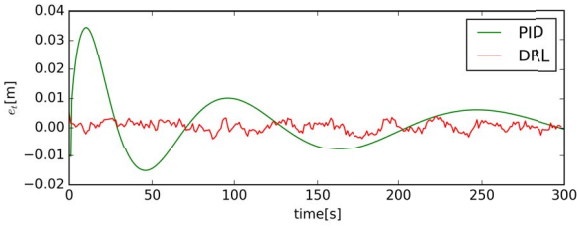

Fig. 5: The total error of DRL(training episode 1000) and PID controller

Figure 7 indicates the change of average reward in relation to the change of the number of training episodes. The reward value converges dramatically and becomes stable after 200 episodes.

In conclusion, after carrying out sufficient trials in reinforcement learning, the trajectory tracking of the DRL shows better performance than that of the PID controller, with higher accuracy and more stability.

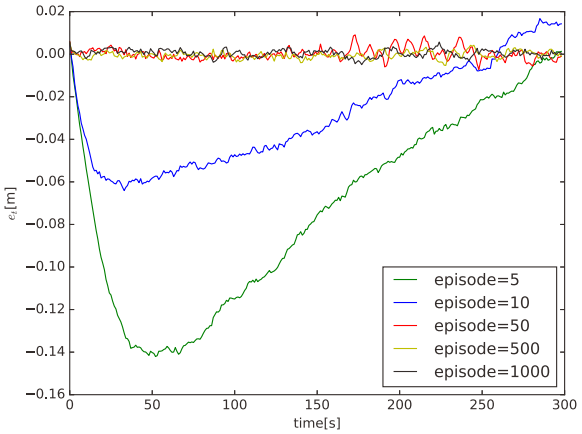

Fig. 6: The total error of DRL in different training episodes

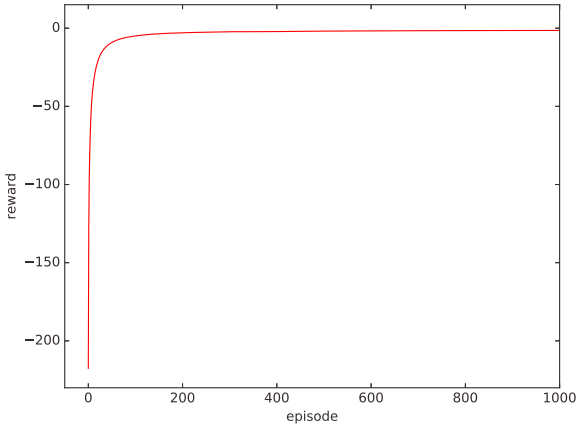

Fig. 7: Total reward

5.2 The simulation results of DRL in complex curve trajectory tracking in a horizontal plane

The desired curve is defined as follows:

$$y_d(t) = 0.5\sin(1.8 \cdot x_d(t))$$

$$x_d(t) = t$$

The new PID parameters are set at: $K_p = 3.01$, $K_I = 4.96$, $K_d = 0.0005$. We set $\eta = [0,0,0]^T$ $v = [0,0,0]^T$ at the beginning of the simulation.

The simulation results are as follows:

Figure 8 shows the trajectory of the DRL (training episode 3000) and PID controller, and figure 9 is the trajectory of the DRL in different training episodes.

Figure 10 shows the total error of DRL(training episode 3000) and PID controller. Figure 11 shows the total error of DRL in different training episodes. It can be seen from the figures that more episodes results in better accuracy in the practical trajectory. The performance of the DRL (episode 3000) is more stable and robust than that of the PID controller.

Figure 12 illustrates the change of average reward in relation to the change of training episodes. It can be seen that the reward value shows an upward trend with significant fluctua-

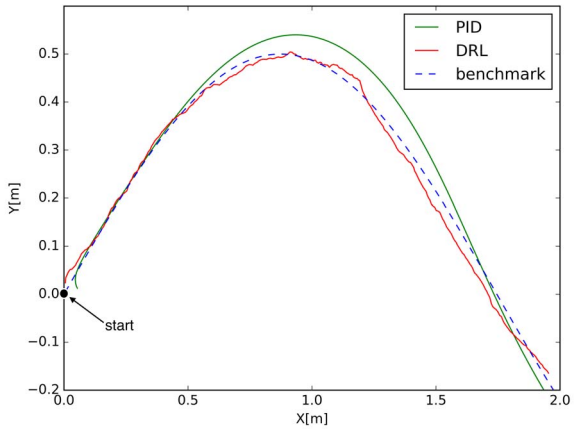

Fig. 8: The trajectory of DRL(training epidode 3000) and PID controller

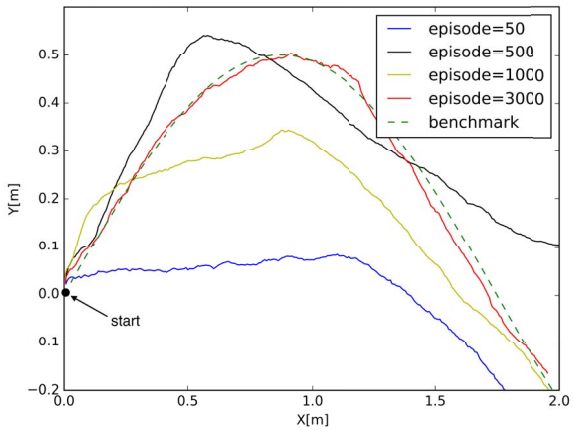

Fig. 9: The trajectory of DRL in different training episodes

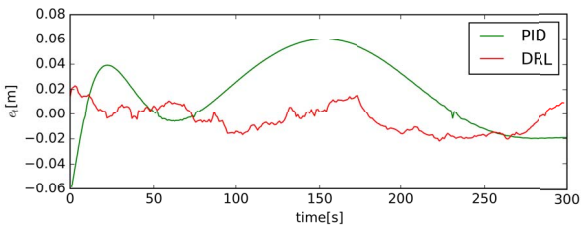

Fig. 10: The total error of DRL(training episode 3000) and PID controller

tion at first but the fluctuation then declines and it eventually stabilizes after 2500 episodes, with a marginal fluctuation.

In conclusion, after carrying out sufficient trials in reinforcement learning, the trajectory tracking of the DRL shows better performance than that of PID controller with higher accuracy and more stability.

6 Conclusions

In this paper, a trajectory tracking control algorithm of AUVs using deep reinforcement learning is developed. Two neural networks are embedded in the algorithm: the actor

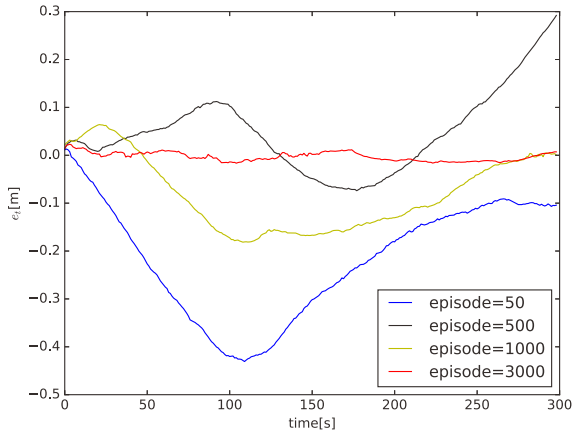

Fig. 11: The total error of DRL in different training episodes

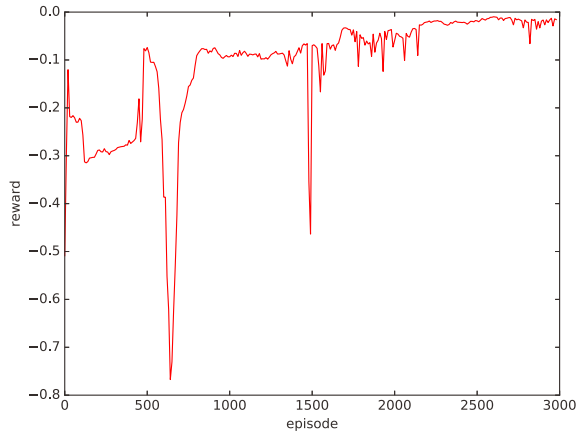

Fig. 12: Total reward

network is used to carry out the control policy while the critic network is used to evaluate the training of actor network. Mathematical proofs have been applied to analyze the stability of the system. In simulations, we verify the robustness and effectiveness of DRL by making comparisons with the PID controller of the performance in different kinds of trajectory tracking. In the future, research should be carried out into the application of the algorithm into practical systems.

7 Appendices

7.1 Underwater dynamic parameters of the system

The dynamic parameters of the system are as follows. Its model is based on Cui's[20].

m_{11}	200kg
m_{12}	250kg
m_{33}	80kg
d_{11}	$(70+100\|u\|)kg/s$
d_{22}	$(100+200\|v\|)kg/s$
d_{33}	$(50+100\|r\|)kg/s$
$g_x(\eta)$	$0N$
$g_y(\eta)$	$0N$
$g_z(\eta)$	$0N$

References

[1] Londhe, Pandurang S., et al. "Task space control of an autonomous underwater vehicle manipulator system by robust single-input fuzzy logic control scheme." *IEEE Journal of Oceanic Engineering* 42.1 (2017): 13-28.

[2] Kim, Min J., et al. "Way-point tracking for a Hovering AUV by PID controller." *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*. IEEE, 2015.

[3] Wang, Lu, et al. "Horizontal tracking control for AUV based on nonlinear sliding mode." *Information and Automation (ICIA), 2012 International Conference on*. IEEE, 2012.

[4] Juan, Li, et al. "AUV control systems of nonlinear extended state observer design." *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*. IEEE, 2014.

[5] De Paula, Mariano, and Gerardo G. Acosta. "Trajectory tracking algorithm for autonomous vehicles using adaptive reinforcement learning." *OCEANS'15 MTS/IEEE Washington*. IEEE, 2015.

[6] El-Fakdi, A., et al. "Autonomous underwater vehicle control using reinforcement learning policy search methods." *Oceans 2005-Europe*. Vol. 2. IEEE, 2005.

[7] Silver, David, et al. "Deterministic policy gradient algorithms." *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014.

[8] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

[9] Fossen, Thor I. *Guidance and control of ocean vehicles*. John Wiley & Sons Inc, 1994.

[10] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. Vol. 1. No. 1. Cambridge: MIT press, 1998.

[11] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks." *Neural networks* 3.5 (1990): 551-560.

[12] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[13] Cui, Rongxin, et al. "Neural network based reinforcement learning control of autonomous underwater vehicles with control input saturation." *Control (CONTROL), 2014 UKACC International Conference on*. IEEE, 2014.

[14] You, Tian Qing, et al. "CFD Research on Underwater Vehicle Hydrodynamic Damping Force Coefficient." *Missiles & Space Vehicles* (2016).

[15] WANG Yao-nan. (1996). Intelligence Control System. Hunan university press. (in Chinese)

[16] Webb, Geoffrey I., Michael J. Pazzani, and Daniel Billsus. "Machine learning for user modeling." *User modeling and user-adapted interaction* 11.1 (2001): 19-29.

[17] Deisenroth, Marc Peter, Gerhard Neumann, and Jan Peters. "A survey on policy search for robotics." *Foundations and Trends in Robotics* 2.12 (2013): 1-142.

[18] Dan, Sven. *Quadratic Programming. Nonlinear and Dynamic Programming*. Springer Vienna, 1975:33-59.

[19] Tobergte, David R, and S. Curtis. "Improving Neural Networks with Dropout." (2013).

[20] Cui, Rongxin, et al. "Leaderfollower formation control of underactuated autonomous underwater vehicles." *Ocean Engineering* 37.17 (2010): 1491-1502.

[21] Ng, Andrew Y. *Shaping and policy search in reinforcement learning*. Diss. University of California, Berkeley, 2003.

[22] Konda, Vijay R., and John N. Tsitsiklis. "Actor-Critic Algorithms." *NIPS*. Vol. 13. 1999.

[23] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature* 521.7553 (2015): 436-444.

[24] Kobayashi, Ryosuke, and Satoshi Okada. "Development of hovering control system for an underwater vehicle to perform core internal inspections." *Journal of Nuclear Science and Technology* 53.4 (2016): 566-573.

[25] Selvakumar, J. Manecius, and T. Asokan. "Station keeping control of underwater robots using disturbance force measurements." *Journal of Marine Science and Technology* 21.1 (2016): 70-85.

[26] Dean, Jeffrey, et al. "Large scale distributed deep networks." *Advances in neural information processing systems*. 2012.