Depth Control of Model-Free AUVs via Reinforcement Learning

Hui Wu, Shiji Song, *Senior Member, IEEE*, Keyou You, *Senior Member, IEEE*, and Cheng Wu

***Abstract*—In this paper, we consider depth control problems of an autonomous underwater vehicle (AUV) for tracking the desired depth trajectories. Due to the unknown dynamical model and the coupling between surge and yaw motions of the AUV, the problems cannot be effectively solved by most of the model-based or proportional-integral-derivative like controllers. To this purpose, we formulate the depth control problems of the AUV as continuous-state, continuous-action Markov decision processes under unknown transition probabilities. Based on the deterministic policy gradient theorem and neural network approximation, we propose a model-free reinforcement learning (RL) algorithm that learns a state-feedback controller from sampled trajectories of the AUV. To improve the performance of the RL algorithm, we further propose a batch-learning scheme through replaying previous prioritized trajectories. We illustrate with simulations that our model-free method is even comparable to the model-based controllers. Moreover, we validate the effectiveness of the proposed RL algorithm on a seafloor data set sampled from the South China Sea.**

***Index Terms*—Autonomous underwater vehicle (AUV), depth control, deterministic policy gradient (DPG), neural network, prioritized experience replay, reinforcement learning (RL).**

I. Introduction

AUTONOMOUS underwater vehicle (AUV) is a type of self-controlled submarine whose flexibility, autonomy, and size-diversity make it advantageous in many applications, including seabed mapping [1], chemical pluming tracing [2], resource gathering, contaminant source localization [3], operation under dangerous environments, maritime rescue, etc. Therefore, the control of AUVs has drawn great attention of the control community. Among many control problems of AUVs, the depth control is crucial in many applications. For example, when performing a seabed mapping, an AUV is required to keep a constant distance from the seafloor.

There are many difficulties for the depth control problems of AUVs. The nonlinear dynamics of AUVs render bad performances of many linear controllers such as linear quadratic Gaussian integral (LQI) and fixed proportional-integral-derivative (PID) controllers. Even with nonlinear controllers, usually it is hard to obtain an exact dynamical model of the AUV in practice. Moreover, the complicated undersea environment brings various disturbances, e.g., sea currents, waves, and model uncertainty, all of which increase the difficulty for the depth control.

Manuscript received July 17, 2017; accepted December 5, 2017. Date of publication January 18, 2018; date of current version November 19, 2019. This work was supported in part by the National Science Foundation of China under Grant 41427806 and Grant 41576101, and in part by the National Key Research and Development Program of China under Grant 2016YFC0300801. This paper was recommended by Associate Editor S. Nahavandi. *(Corresponding author: Shiji Song.)*

The authors are with the Department of Automation and the TNList, Tsinghua University, Beijing 100084, China (e-mail: wuhui115199@163.com; shijis@tsinghua.edu.cn; youky@tsinghua.edu.cn; wuc@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2017.2785794

Most of the conventional control approaches mainly focus on solving the control problems of the AUV based on an exact dynamical model. An extended PID controller including an acceleration feedback is proposed for the dynamical positioning systems of marine surface vessels [4]. The controller compensates the disturbances of the slow-varying forces by introducing a measured acceleration feedback. A self-adaptive PID controller tuned by Mamdani fuzzy rule is used to control a nonlinear AUV system in tracking the heading and depth with a stabilized speed, and outperforms classical tuned PID controllers [5].

Other model-based controllers of AUVs include backstepping [6], [7], sliding-mode [8], [9], model predictive control (MPC) [10], [11], robust control [12], etc. An adaptive backstepping controller is designed for the tracking problems of ships and mechanical systems which guarantees uniform global asymptotic stability of the closed-loop tracking error [13]. Combined with line-of-sight guidance, two sliding mode controllers are, respectively, designed for the sway-yaw-roll control of a ship [8]. MPC is a control strategy where control input at each sampling time is estimated based on the predictions over a certain horizon [14]. In [15], a controller is proposed by solving a particular MPC problem and controls a nonlinear constrained submarine to track the sea floor. In [16], nonlinear depth and yaw controllers are proposed based on the robust control for set-point regulation and trajectory tracking of the L2ROV underwater vehicle. The proposed controllers are easy tuning for real applications and proved stable by Lyapunov arguments.

However, the performance of model-based controllers will seriously degrade under an incorrect dynamical model. In practical applications, the accurate models of AUVs are difficult to obtain due to the complex undersea environment. For such a case, a model-free controller is required, which is learned by reinforcement learning (RL) in this paper.

RL is a dynamic-programming-based solving framework for the Markov decision process (MDP) without the transition model. It has been applied successfully in robotic control problems, including the path planning for a single mobile robot [17] or multirobot [18], robot soccer [19], biped robot [20], unmanned aerial vehicle (UAV) [21], etc.

In this paper, we propose an RL framework for the depth control problems of the AUV based on the deterministic policy gradient (DPG) theorem and the neural network approximation. We consider three depth control problems, including constant depth control, curved depth tracking, and seafloor tracking according to different forms and observable information of target trajectories.

The key of applying RL is how to model the depth control problems as MDPs. MDP describes such a process where an agent at some state takes an action and transits to the next state with an one-step cost. In our problems, the definitions of the "state" and "one-step cost" are significant for the performance of the RL. Usually the motions of an AUV are described by six coordinates and their derivatives. It is straightforward to regard the coordinates as the states of MDPs directly. However, this scenario is not perfect for excluding the information of the target depth trajectories. In addition, the angle variables in the motion coordinates are periodic thus cannot be used as components of the state directly. Therefore, one of our works is designing a better state to overcome these defects.

Most of the RL algorithms usually approximate a value function and a policy function both of which are used to evaluate and generate a policy, respectively, while the forms of the approximators depend on the transition model of the MDP. For the depth control problems of the AUV, the nonlinear dynamics and the constrained control inputs result in serious approximation difficulties. In this paper, we choose neural network approximators because of their powerful representation abilities. We consider how to design adaptive structures of the networks for the depth problems of AUVs.

After designing the networks, we train them based on the sampled trajectories of the AUV, which compensates the model-free limitation. However, the capacity of battery and storage of an ocean-going AUV limits the volume of sampled data during undersea tasks, thus the sampled trajectories are usually insufficient for the training demand. To improve the data efficiency, we proposed a batch-learning scheme through replaying previous experiences.

The main contributions of this paper are summarized as follows.

1) We formulate three depth control problems of AUVs as MDPs with well-designed states and cost functions.
2) We design two neural networks with specific structures and rectified linear unit (ReLu) activation functions.
3) We propose a batch-gradient updating scheme called prioritized experience replay and incorporate it into our RL framework.

The remainder of this paper is organized as follows. In Section II, we describe the motions of AUVs and the three depth control problems. In Section III, the depth control problems are modeled as MDPs under well-designed states and one-step cost functions. In Section IV, an RL algorithm based on the DPG is applied to solve the MDPs. In Section V, we highlight several innovative techniques which are proposed to improve the performance of the RL algorithm. In Section VI, the simulations are performed on a classic REMUS AUV and the performances of two model-based controllers are compared with that of our algorithm to validate its effectiveness. In addition, an experiment on a real seafloor data set is performed to show the practicability of our RL framework.

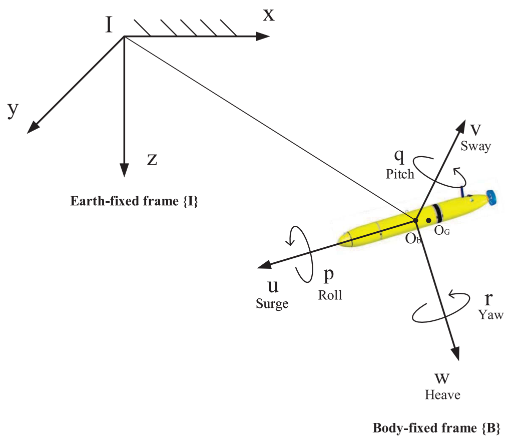


Fig. 1. Two coordinate frames and six DOFs motions of the AUV.

II. Problem Formulation

In this section, we describe the coordinate frames of AUVs and the depth control problems.

A. Coordinate Frames of AUVs

The motions of an AUV have six degrees of freedom (DOFs) including surge, sway, and heave, which refer to the longitudinal, sideways, and vertical displacements, and yaw, roll, and pitch, which describe the rotations around the vertical, longitudinal, and transverse axes. Fig. 1 illustrates the details of the six DOFs.

Correspondingly there are six independent coordinates determining the position and orientation of the AUV. An earth-fixed coordinate frame $\{\boldsymbol{I}\}$ is defined for the six coordinates corresponding to the position and the orientation along the x, y, and z axes denoted by $\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^T$. The earth-fixed frame is assumed to be inertial by ignoring the effect of the earth's rotation. The linear and angular velocities denoted by $\boldsymbol{\nu} = [u, v, w, p, q, r]^T$ are described in a body-fixed coordinate frame $\{\boldsymbol{B}\}$, which is a moving coordinate frame whose origin is fixed to the AUV. Fig. 1 shows the two coordinate frames and six coordinates.

B. Depth Control Problems

For simplicity, we just consider the depth control problems of AUVs on the x–z plane in this paper, all of which can be easily extended to 3-D cases. Thus we only examine the motions on the x–z plane and drop the terms out of the plane. Furthermore, the surge speed u is assumed to be constant. The remaining coordinates are denoted by the vector $\boldsymbol{\chi} = [z, \theta, w, q]^T$, including heave position z, heave velocity w, pitch orientation θ, and pitch angular velocity q.

The dynamical equation of the AUV is defined as follows:

$$\dot{\boldsymbol{\chi}} = f(\boldsymbol{\chi}, \boldsymbol{u}, \boldsymbol{\xi}) \tag{1}$$

where $\boldsymbol{u}$ denotes the control vector and $\boldsymbol{\xi}$ denotes the possible disturbance.

The purpose of the depth control is to control the AUV to track a desired depth with minimum energy consumption, where the desired depth trajectory z_r is given by

$$z_r = g(x). \tag{2}$$

We are concerned with the depth control under three cases according to the form and the information about z_r.

1) *Constant Depth Control:* The constant depth control is to control the AUV to operate at a constant depth. The desired depth is constant, i.e., $\dot{z}_r = 0$. To prevent the AUV from oscillating around z_r, the heading direction is required to keep consistent with the x-axis, i.e., the pitch angle $\theta = 0$.
2) *Curved Depth Tracking:* The curved depth tracking is to control the AUV to track a given curved depth trajectory. The desired depth trajectory is a curve and its derivative $\dot{z}_r$ that describes the trend of the curve is known. Besides minimizing $|z - z_r|$, the pitch angle is required to keep consistent with the slope angle of the curve.
3) *Seafloor Tracking:* Seafloor tracking is to control the AUV to track the seafloor and keep a constant safe distance simultaneously. When tracking the seafloor, the AUV can only measure the relative depth $z - z_r$ from itself to the seafloor using sonar-like devices, thus the slope angle of the seafloor curve is unknown. The missing information increases the difficulty of the control problem compared with the other two cases.

III. MDP Modeling

In this section, we model the above three depth control problems as MDPs with unknown transition probabilities due to the unknown dynamics of the AUV.

A. Markov Decision Process

An MDP is a stochastic process satisfying the Markov property. It consists four components: 1) a state space $\mathcal{S}$; 2) an action space $\mathcal{A}$; 3) a one-step cost function $c(\boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and 4) a stationary one-step transition probability $p(\boldsymbol{s}_t|\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$. The Markov property means that current state is only dependent on the last state and action, i.e.,

$$p(\boldsymbol{s}_t|\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}) = p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}). \tag{3}$$

The MDP describes how an agent interacts with the environment: at some time step t, the agent in state $\boldsymbol{s}_t$ takes an action $\boldsymbol{a}_t$ and transfers to the next state $\boldsymbol{s}_{t+1}$ according to the transition probability with an observed one-step cost $c_t = c(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Fig. 2 illustrates the evolution of the MDP.

The MDP problem is to find a policy to minimize the long-term cumulative cost function. The policy is a map from the state space $\mathcal{S}$ to the action space $\mathcal{A}$, and can be defined as a function form $\pi : \mathcal{S} \to \mathcal{A}$ or a distribution $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Therefore, the optimization problem is formulated as

$$\min_{\pi\in\mathcal{P}} J(\pi) = \min_{\pi\in\mathcal{P}} E\left[\sum_{k=1}^{K} \gamma^{k-1} c_k|\pi\right] \tag{4}$$

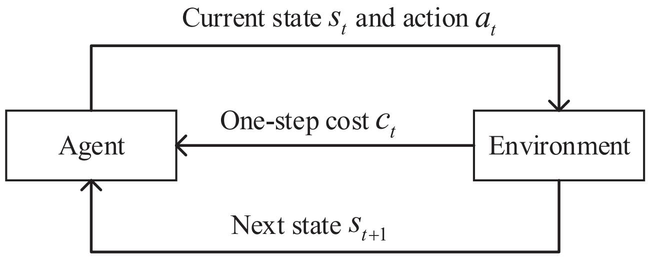


Fig. 2. Evolution of MDP.

where $\mathcal{P}$ denotes the policy space and γ is a discounted factor with $0 < \gamma < 1$. The superscript K of the summation represents the horizon of the problem.

The definitions of four components of MDPs are essential for the performances of RL algorithms. For the depth control problems of the AUV, the unknown dynamical model means the unknown transition probability, and the actions correspond to the control inputs, i.e., $\boldsymbol{a} = \boldsymbol{u}$. Therefore, the key is how to design the states and the one-step cost functions for the control problems.

B. MDP for Constant Depth Control

The purpose of the constant depth control problem is to control the AUV to operate at a constant depth z_r. We design a one-step cost function of the form

$$c(\boldsymbol{\chi}, \boldsymbol{u}) = \rho_1(z - z_r)^2 + \rho_2\theta^2 + \rho_3 w^2 + \rho_4 q^2 + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} \tag{5}$$

where $\rho_1(z - z_r)^2$ is for minimizing the relative depth, $\rho_2\theta^2$ for keeping the pitch angle along the x-axis, and the last three terms for minimizing the consumed energy. The cost function provides a tradeoff between different controlling objectives through the coefficients ρ_1–ρ_4.

It is intuitive to choose $\boldsymbol{\chi}$ which describes the motions of the AUV as the state. However, this choice performs worse in practice for two reasons. First, the pitch angle θ, an angular variable, cannot be added to the state directly due to its periodicity. For example, a state with $\theta = 0$ and one with $\theta = 2\pi$ seem different but actually equivalent with each other. Hence we divide the pitch angle into two trigonometric components $[\cos(\theta), \sin(\theta)]^T$ to eliminate the periodicity.

The second drawback is the absolute depth z in the state. Suppose that if we have learned a controller to keep the AUV at a specific depth z_r, and now the target depth changes to a new depth z_r' that the AUV has never visited. The controller for the old z_r will not work for the new depth which is not covered by the state space explored before. A better choice is the relative depth $\Delta z \doteq z - z_r$ which denotes how far the AUV is from the target depth. A optimal controller learned with the state containing Δz will be aware of controlling the AUV to rise or dive when Δz is negative or positive.

To overcome the above drawbacks, we design the state of the constant depth control as follows:

$$\boldsymbol{s} = \left[\Delta z, \cos(\theta), \sin(\theta), w, q\right]^T. \tag{6}$$

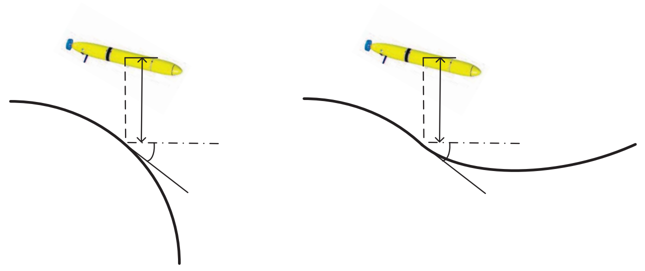


Fig. 3. In these two situations, the AUVs in the same state cannot track the two curves under the same controller when the state is defined as (6) because of the different $\dot{\theta}_c$ of the curves.

C. MDP for Curved Depth Control

The curved depth control is to control the AUV to track a given curved depth trajectory $z_r = g(x)$. The state (6) is insufficient for the curved depth control because it does not include the future trend of the target depth denoted by the slope angle θ_c and its derivative $\dot{\theta}_c$ of the curve. The slope angle θ_c helps to anticipate whether the AUV should rise or dive, and $\dot{\theta}_c$ represents the change of rate of θ_c.

Fig. 3 shows two situations where the AUVs in the same state defined as (6) are tracking two curves with different $\dot{\theta}_c$. Obviously, they cannot be controlled to track the two curves under the same policy. The failure results from that the AUV cannot anticipate the trend of the curve.

In order to add the information about $\dot{\theta}_c$ to the state, we consider its form

$$\dot{\theta}_c = \frac{g''(x)}{\left[1+g'(x)\right]^2}\dot{x} = \frac{g''(x)}{[1+g'(x)]^2}(u_0\cos\theta + w\sin\theta) \quad (7)$$

where u_0 denotes the constant surge speed of the AUV, and $g'(x)$ and $g''(x)$ denote the first and second derivative of $g(x)$ with respect to x.

Then we consider the derivative of the relative depth $z_\Delta \doteq z - z_r$ as follows:

$$\begin{aligned}\dot{z}_\Delta &= \dot{z} - \dot{z}_r\\ &= w\cos\theta - u_0\sin\theta + (u_0\cos\theta + w\sin\theta)\tan\theta_c\\ &= \frac{w\cos(\theta-\theta_c) - u_0\sin(\theta-\theta_c)}{\cos\theta_c}\\ &= \frac{w\cos\theta_\Delta - u_0\sin\theta_\Delta}{\cos\theta_c}\end{aligned} \quad (8)$$

where $\theta_\Delta \doteq \theta - \theta_c$ denotes the relative heading angle between the AUV and the target depth curve. Inspired by the form of $\dot{z}_\Delta$, we design the following one-step cost function:

$$c(z_\Delta, \theta_\Delta, w, q, \boldsymbol{u}) = \rho_1 z_\Delta^2 + \rho_2\theta_\Delta^2 + \rho_3 w^2 + \rho_4 q^2 + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} \quad (9)$$

where $\rho_1 z_\Delta^2$ and $\rho_2\theta_\Delta^2$ minimize the relative depth and the relative heading direction, respectively. It means that the AUV is controlled to track the depth curve and its trend simultaneously.

The state for the curved depth control is defined as

$$\boldsymbol{s} = \left[z_\Delta, \cos\theta_\Delta, \sin\theta_\Delta, \cos\theta_c, \sin\theta_c, \dot{\theta_\Delta}, w, q\right]^T. \quad (10)$$

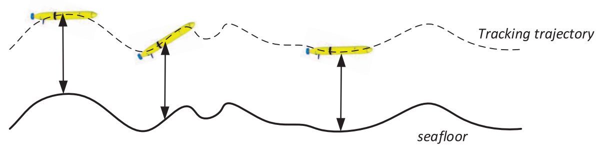


Fig. 4. Seafloor tracking problem.

D. MDP for Seafloor Tracking

Illustrated in Fig. 4, seafloor tracking is to control the AUV to track the seafloor while keeping a constant relative depth in case of colliding. In such a case the AUV can only measure the relative vertical distance $\Delta z = z - z_r$ from the seafloor by a sonar-like device, but cannot observe the slope angle θ_c and its derivative $\dot{\theta}_c$ of the seafloor curve.

Therefore, the state (10) is no longer feasible due to the missing θ_c and $\dot{\theta}_c$. If we adopt the state defined in (6), the problem illustrated in Fig. 3 still exists. Actually, this problem is also called "perceptual aliasing" [22] which means that different parts of the environment appear similar to the sensor system of the AUV. The reason is that the state (6) is just a partial observation of the environment. Thus we consider expanding the state to contain more information.

Though unmeasured by the AUV, the trend of the seafloor curve can still be estimated by the most recent observed sequence of the relative vertical distances, i.e., $[\Delta z_{t-N+1}, \ldots, \Delta z_{t-1}, \Delta z_t]$, where N denotes the length of the sequence. Therefore, with the same one-step cost function (5), we define the expanded state for the seafloor tracking problem as

$$\boldsymbol{s} = \left[\Delta z_{t-N+1}, \ldots, \Delta z_{t-1}, \Delta z_t, \cos\theta, \sin\theta, w, q\right]^T. \quad (11)$$

The length of the recently measured sequence N is significant for the performance of the state, so we will discuss the best setting for N in the simulation.

IV. Solving MDPs of Depth Control via RL

In this section, we adopt the RL algorithm to solve the MDPs for the depth control problems of the AUV in Section III.

A. Dynamic Programming

Here we introduce a classic solving routine for MDPs called *dynamic programming* as the basis of our RL algorithm for the depth control. We first define two types of functions that evaluate the performance of a policy for convenience. *Value function* is a long-term cost function defined by

$$V^\pi(\boldsymbol{s}) = E\left[\sum_{k=1}^{K}\gamma^{k-1}c_k|\boldsymbol{s}_1 = \boldsymbol{s}, \pi\right] \quad (12)$$

with a starting state $\boldsymbol{s}_1$ under a specific policy π. *Action-value* function (also called the Q-value function) is a value function

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = E\left[\sum_{k=1}^{K}\gamma^{k-1}c_k|\boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}, \pi\right] \quad (13)$$

with a chosen starting action $\boldsymbol{a}_1$.

Note that the relationship between the long-term cost function (4) and the value function $V^{\pi}(\boldsymbol{s})$ is given as

$$J(\pi) = \int_{\boldsymbol{s}} p_1(\boldsymbol{s}) V^{\pi}(\boldsymbol{s}) d\boldsymbol{s} \tag{14}$$

where $p_1(\boldsymbol{s})$ denotes the initial state distribution. Thus the minimization (4) is equivalent with the *Bellman optimality equation* for every state

$$\begin{aligned} V^*(\boldsymbol{s}) &= \min_{\pi \in \mathcal{P}} V^{\pi}(\boldsymbol{s}) \\ &= \min_{\pi \in \mathcal{P}} \int_{\boldsymbol{a}} \pi(\boldsymbol{a}|\boldsymbol{s}) \cdot \left[c(\boldsymbol{s}, \boldsymbol{a}) + \int_{\boldsymbol{s}'} p\left(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}\right) V^*\left(\boldsymbol{s}'\right) d\boldsymbol{s}' \right] d\boldsymbol{a}. \end{aligned} \tag{15}$$

The Bellman optimality equation determines the basic routine of solving the MDP problem comprising two phases known as *policy evaluation* and *policy improvement* [23]. The policy evaluation estimates the value function of a policy π by iteratively using the Bellman equation

$$V_{j+1}^{\pi}(\boldsymbol{s}) = \int_{\boldsymbol{a}} \pi(\boldsymbol{a}|\boldsymbol{s}) \left[c(\boldsymbol{s}, \boldsymbol{a}) + \int_{\boldsymbol{s}'} p\left(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}\right) V_j^{\pi}\left(\boldsymbol{s}'\right) d\boldsymbol{s}' \right] d\boldsymbol{a} \tag{16}$$

with an initially supposed value function $V_0^{\pi}(\boldsymbol{s})$. The iteration can be performed until the convergence (policy iteration) or for fixed steps (generalized policy iteration), or even one step (value iteration) [23]. After the policy evaluation, the policy improvement takes place to obtain an improved policy based on the estimated value function by a greedy minimization

$$\pi'(\boldsymbol{s}) = \operatorname*{argmin}_{\boldsymbol{a} \in \mathcal{A}} \left[c(\boldsymbol{s}, \boldsymbol{a}) + \int_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) V^{\pi}(\boldsymbol{s}') d\boldsymbol{s}' \right]. \tag{17}$$

Both phases are iterated alternatively until the policy converges to the optimal one.

B. Temporal Difference and Value Function Approximation

The dynamic programming only fits to the MDPs with finite state and finite action spaces under a known transition probability. For the MDPs constructed in Section III, the transition probabilities $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ are unknown due to the unknown dynamics of the AUV which is required by the value function updating (16). Thus we adopt a new rule known as temporal difference (TD) to update the value function using sampled transition data.

Suppose observing a transition pair $(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})$ along a trajectory of the AUV at time k, then TD updates the Q-value function as the form [23]

$$\begin{aligned} Q(\boldsymbol{s}_k, \boldsymbol{u}_k) \leftarrow{} & Q(\boldsymbol{s}_k, \boldsymbol{u}_k) \\ & + \alpha \left[c(\boldsymbol{s}_k, \boldsymbol{u}_k) + \gamma \min_{\boldsymbol{u}} Q(\boldsymbol{s}_{k+1}, \boldsymbol{u}) - Q(\boldsymbol{s}_k, \boldsymbol{u}_k) \right] \end{aligned}$$

where $\alpha > 0$ is a learning rate.

The TD algorithm updates the map from state-action pairs to their Q values and stores it as a lookup table. However, for the depth control problems, the states consist of the motion vector $\boldsymbol{\chi}$ of the AUV and the desired depth z_r, while the actions are usually forces and torques of the propellers. All these continuous variables lead to continuous state and action spaces which cannot be represented by a lookup table.

We represent the map by a parameterized function $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$ and instead update the parameter $\boldsymbol{\omega}$ as follows:

$$\begin{aligned} \delta_k &= c(\boldsymbol{s}_k, \boldsymbol{u}_k) + \gamma Q(\boldsymbol{s}_{k+1}, \mu(\boldsymbol{s}_{k+1}|\boldsymbol{\theta})|\boldsymbol{\omega}) - Q(\boldsymbol{s}_k, \boldsymbol{u}_k|\boldsymbol{\omega}) \\ \boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k + \alpha \delta_k \nabla_{\boldsymbol{\omega}} Q(\boldsymbol{s}_k, \boldsymbol{u}_k|\boldsymbol{\omega}) \end{aligned} \tag{18}$$

where $\mu(\boldsymbol{s}_{k+1}|\boldsymbol{\theta})$ is a policy function defined in the next section.

C. Deterministic Policy Gradient

The continuous control inputs for the depth control of the AUV result in continuous actions, thus the minimization (17) over the continuous action space is time-consuming if executed every iteration.

Instead, we implement the policy improvement phase by the DPG algorithm. The DPG algorithm assumes a deterministic parameterized policy function $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ and updates the parameter $\boldsymbol{\theta}$ along the negative gradient of the long-term cost function

$$\boldsymbol{\theta}_{k+1} \doteq \boldsymbol{\theta}_k - \alpha \widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} \tag{19}$$

where $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$ is a stochastic approximation of the true gradient. The DPG algorithm derives the form of $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$ [24] as follows:

$$\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} = \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \mu(\boldsymbol{s}_i|\boldsymbol{\theta}) \nabla_{\boldsymbol{u}_i} Q^{\mu}(\boldsymbol{s}_i, \boldsymbol{u}_i) \tag{20}$$

where Q^{μ} denotes the Q-value function under the policy $\mu(\boldsymbol{u}|\boldsymbol{s})$.

In the last section, the Q-value function is approximated by a parameterized approximator $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$, thus we replace $\nabla_{\boldsymbol{u}_i} Q^{\mu}(\boldsymbol{s}_i, \boldsymbol{u}_i)$ with $\nabla_{\boldsymbol{u}_i} Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega})$. The approximate gradient is given by

$$\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} \approx \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \mu(\boldsymbol{s}_i|\boldsymbol{\theta}) \nabla_{\boldsymbol{u}_i} Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega}). \tag{21}$$

Note that the approximate gradient in (21) is calculated by a transition sequence, which seems to be inappropriate with the online characteristic of the depth control problems. We can still get an online updating rule if setting $M = 1$, but the deviation of the approximation may be amplified. Actually, a batch updating scheme can be performed by sliding the sequence along the trajectory of the AUV.

We have defined two function approximators $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$ and $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ in TD and DPG algorithms, but do not give the detailed forms of the approximators. Because of the nonlinear and complicated dynamics of the AUV, we constructed two neural network approximators, evaluation network $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$ and policy network $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ with $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ as the weights.

To illustrate the updation of the two networks through TD and DPG algorithms, we present a structure diagram shown in Fig. 5. The ultimate goal of the RL algorithm is to learn the state feedback controller represented by the policy network. There are two back-propagating paths in our algorithm. The evaluation network is back-propagated by the error between the current Q-value $\hat{Q}(\boldsymbol{s}_k, \boldsymbol{u}_k)$ and that of successor state-action pair $\hat{Q}(\boldsymbol{s}_{k+1}, \boldsymbol{u}_{k+1})$ plus the one-step cost, which is

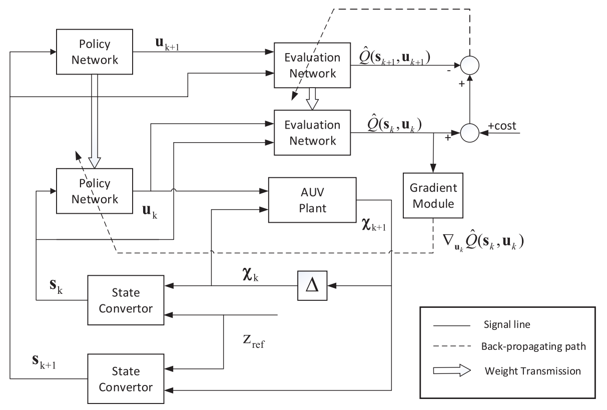


Fig. 5. Structure diagram of NNDPG.

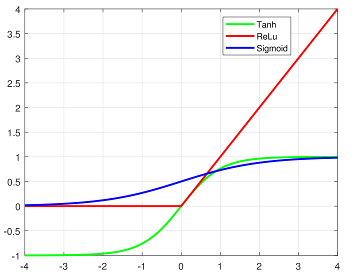


Fig. 6. Curves of ReLu, sigmoid, and tanh functions, where the ReLu only inhibits changes along the negative x-axis and the other two inhibit changes in two directions.

the idea of the TD algorithm. The output of the evaluation network is passed to the "gradient module" to generate the gradient $\nabla_{\boldsymbol{u}_k}\hat{Q}(\boldsymbol{s}_k, \boldsymbol{u}_k)$, which is then propagated back to update the policy network through the DPG algorithm. The two back-propagating paths correspond, respectively, to the policy evaluation and policy improvement phases in dynamic programming. Note that the module "state convertor," the process of state-designing illustrated in Section III, transforms the AUV's coordinates $\boldsymbol{\chi}_k$ and reference depth signal z_{ref} into the state $\boldsymbol{s}_k$.

V. Improved Strategies

In the last section, we have illustrated an RL framework for the depth control of the AUV, which updates two neural network approximators by iterating the TD and DPG algorithms. Combining the characteristics of the control problems, we further proposed improved strategies from two aspects. First, we design adaptive structures for the neural networks according to the physical constrains when controlling the AUV, where a new type of activation function is adopted. Then we propose a batch-learning scheme to improve the data-efficiency.

A. Neural Network Approximators

Considering the nonlinear dynamics of the AUV, we construct two neural network approximators, evaluation network $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$ and policy network $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ with $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ as the weights.

The evaluation network has four layers with the state $\boldsymbol{s}$ and the control variable $\boldsymbol{u}$ as the inputs where $\boldsymbol{u}$ is not included until the second layer referring to [25]. The output layer is a linear unit to generate a scalar Q-value.

The policy network is designed with three layers and generates the control variable $\boldsymbol{u}$ given the inputting state $\boldsymbol{s}$. Due to the limited power of the propellers of the AUV, the output $\boldsymbol{u}$ must be constrained in a given range. Therefore, we adopt tanh units as the activation functions of the output layer. The output of the tanh function in $[-1, 1]$ is then scaled to the given interval.

Besides the structures of the networks, we adopt a new type of activation function known as ReLu for the hidden layers.

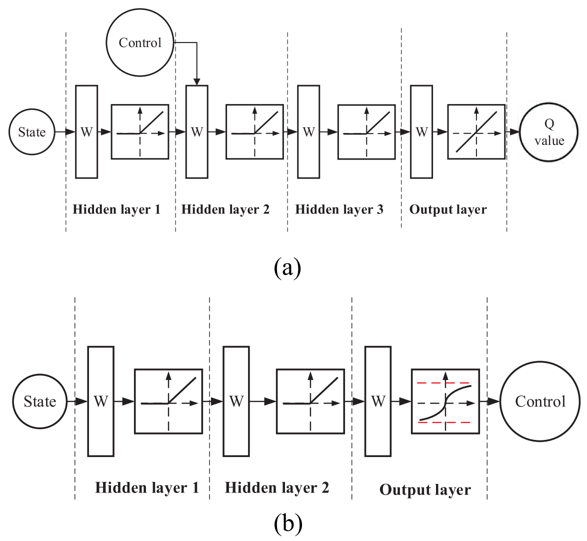


Fig. 7. Structures of two neural networks. (a) Evaluation network has four layers with the control input included at the second layer. (b) Policy network has three layers with tanh function as the output activation unit.

In traditional neural network controllers, the activation functions usually use sigmoid or tanh functions which are sensitive to inputs near zero but insensitive to large ones. This saturate property brings a "gradient vanish" problem where the gradient of the unit reduces to zero under a large input and does not help the training of the network. However, the ReLu function avoids the problem as it only inhibits changes in one direction as illustrated in Fig. 6. Moreover, the simpler form of the ReLu can accelerate the calculation of the network, which just fits the online property of the depth control for the AUV.

In the end, we illustrate the full structures of the two networks in Fig. 7.

B. Prioritized Experience Replay

In this section, we consider how to improve the data efficiency for the depth control problems of the AUV. Instead of designing the controller based on an accurate model, our RL algorithm learns the optimal policies from the transition records which are sampled along the trajectories of the AUV during underwater tasks. However, the battery and memory capacity of an ocean-going AUV limits the volume of sampled data for each undersea task. Recharging and deploying the AUV for multiple tasks are consuming and risky because of the complex undersea environment. Therefore, it is essential for our RL algorithm to learn the optimal controller from a finite amount of sampled data.

We propose a batch learning scheme called *prioritized experience replay* which is an improved version of the *experience replay* proposed by Lin [26]. Imagine the scene of controlling the AUV with the RL algorithm. The AUV observes the subsequent state $\boldsymbol{s}'$, one-step cost c in state $\boldsymbol{s}$ with control input $\boldsymbol{u}$ executed for each time. We call a transition $(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}', c)$ as an "experience." Instead of updating the evaluation network and policy network by the newly sampled experience, the experience replay uses a cache to store all the visited experiences and samples a batch of previous experiences from the cache to update the two networks. The replay mechanism reuses previous experiences as if they were visited newly, which improves the data efficiency greatly.

However, not all experiences should be focused equally. If an experience brings minor differences to the weights of the networks, it does not deserve to be replayed since the networks have learned its implicit pattern. Whereas a "wrong" experience should be replayed frequently.

Inspired by (18), the prioritized experience replay adopts the "error" of the TD algorithm as the priority of an experience, which is given by

$$\text{PRI}_k = |c(\boldsymbol{s}_k, \boldsymbol{u}_k) + \gamma Q(\boldsymbol{s}_{k+1}, \mu(\boldsymbol{s}_{k+1}|\boldsymbol{\theta})|\boldsymbol{\omega}) - Q(\boldsymbol{s}_k, \boldsymbol{u}_k|\boldsymbol{\omega})|. \tag{22}$$

The priority measures how probable the experience is sampled from the cache. Therefore, an experience with higher priority makes more differences to the weight of the evaluation network and is replayed with greater probability.

To sum up, we propose an algorithm called neural-network-based DPG (NNDPG) by combining the above techniques. A more detailed procedure of NNDPG is given in Algorithm 1.

In the end of this section, it is necessary to discuss the advantages of NNDPG. First, NNDPG does not require any knowledge about the AUV model but can still learn a controller whose performance is competitive with the controllers under the exact dynamics. What is more, it greatly improves the data efficiency and the performance by proposing the prioritized experience replay which is first used in the RL controller for the control problems of the AUV.

Algorithm 1 NNDPG for the Depth Control of the AUV

```
Input:
    Target depth $z_r$, number of episodes $M$, number of steps for each episode $T$, batch size $N$, learning rates for evaluation and policy networks $\alpha_{\boldsymbol{\omega}}$ and $\alpha_{\boldsymbol{\theta}}$.
Output:
    Depth control policy $\boldsymbol{u} = \mu(\boldsymbol{s}|\boldsymbol{\theta})$.
1:  Initialize evaluation network $Q(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\omega})$ and policy network $\mu(\boldsymbol{s}|\boldsymbol{\theta})$. Initialize the experience replay cache $R$.
2:  for $i = 1$ to $M$ do
3:      Reset the initial state $\boldsymbol{s}_0$.
4:      for $t = 0$ to $T$ do
5:          Generate a control input $\boldsymbol{u}_t = \mu(\boldsymbol{s}_t|\boldsymbol{\theta}) + \triangle\boldsymbol{u}_t$ with current policy network and exploration noise $\triangle\boldsymbol{u}_t$.
6:          Execute control $\boldsymbol{u}_t$ and obtain $\boldsymbol{s}_{t+1}$. Calculate one-step cost $c_t$ according to (5) or (9).
7:          Push transition tuple $(\boldsymbol{s}_t, \boldsymbol{u}_t, c_{t+1}, \boldsymbol{s}_{t+1})$ into $R$ and calculate its priority according to (22)
8:          Sample a minibatch of $N$ transitions $\{(\boldsymbol{s}_i, \boldsymbol{u}_i, c_{i+1}, \boldsymbol{s}_{i+1}) \mid 1 \le i \le N\}$ from $R$ according to their priorities.
9:          Update evaluation network and refresh the priorities of samples.
                $y_i = c_{i+1} + \gamma Q(\boldsymbol{s}_{i+1}, \mu(\boldsymbol{s}_{i+1}|\boldsymbol{\theta})|\boldsymbol{\omega})$
                $\delta_i = y_i - Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega})$
                $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \frac{1}{N}\alpha_{\boldsymbol{\omega}} \sum_{i=1}^{N} \delta_i \nabla_{\boldsymbol{\omega}} Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega})$
10:         Compute $\nabla_{\boldsymbol{u}_i} Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega})$ for each sample.
11:         Update policy network by
                $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{1}{N}\alpha_{\boldsymbol{\theta}} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \mu(\boldsymbol{s}_i|\boldsymbol{\theta}) \nabla_{\boldsymbol{u}_i} Q(\boldsymbol{s}_i, \boldsymbol{u}_i|\boldsymbol{\omega})$
12:     end for
13: end for
```

VI. Performance Study

A. AUV Dynamics for Simulation

In this section, we present a set of explicit dynamical equations of a classical a six-DOF "REMUS" AUV model [27], which is utilized to validate our algorithm. However, the experiments can be extended easily to other AUV models since our algorithm is totally model-free.

As mentioned, we only consider the motions of the AUV in the x–z plane and assume a constant surge speed u_0. The simplified dynamical equations are given by

$$\begin{aligned} &m\Big(\dot{w} - uq - x_G\dot{q} - z_G q^2\Big) \\ &\quad = Z_{\dot{q}}\dot{q} + Z_{\dot{w}}\dot{w} + Z_{uq}uq + Z_{uw}uw + Z_{ww}w|w| \\ &\qquad + Z_{qq}q|q| + (W - B)\cos\theta + \tau_1 + \triangle\tau_1 \end{aligned} \tag{23a}$$

$$\begin{aligned} &I_{yy}\dot{q} + m\big[x_G(uq - \dot{w}) + z_G wq\big] \\ &\quad = M_{\dot{q}}\dot{q} + M_{\dot{w}}\dot{w} + M_{uq}uq + M_{uw}uw + M_{ww}w|w| \\ &\qquad + M_{qq}q|q| - (x_G W - x_B B)\cos\theta \\ &\qquad - (z_G W - z_B B)\sin\theta + \tau_2 + \triangle\tau_2 \end{aligned} \tag{23b}$$

$$\dot{z} = w\cos\theta - u\sin\theta \tag{23c}$$

$$\dot{\theta} = q \tag{23d}$$

where $[x_G, y_G, z_G]^T$ and $[x_B, y_B, z_B]^T$ are, respectively, the centers of gravity and buoyancy; $Z_{\dot{q}}$, $Z_{\dot{w}}$, $M_{\dot{q}}$, and $M_{\dot{w}}$ denote the added masses; Z_{uq}, Z_{uw}, M_{uq}, and M_{uw} denote the body lift force and moment coefficients; Z_{ww}, Z_{qq}, M_{ww}, and M_{qq} the cross-flow drag coefficients; and W and B represent the ROV's weight and buoyancy. The bounded control inputs τ_1 and τ_2 are propeller thrusts and torques with disturbances $\triangle\tau_1$ and $\triangle\tau_2$ caused by unstable underwater environment. The values of hydrodynamic coefficients are presented in Table I. Notably, the equations contain the coupling terms in the surge and yaw motions of the AUV.

B. Linear Quadratic Gaussian Integral Control

We compare two model-based controllers with the state feedback controller learned by NNDPG. The first is the linear quadratic Gaussian integral (LQI) controller [28] derived from a linearized AUV model.

TABLE I
HYDRODYNAMIC PARAMETERS OF THE REMUS AUV

Coefficients	Value	Coefficients	Value
$m(\text{kg})$	30.51	$I_{yy}(\text{kg}\cdot\text{m}^2)$	3.45
$M_{\dot{q}}(\text{kgm}^2/\text{rad})$	-4.88	$M_{ww}(\text{kg})$	3.18
$M_{\dot{w}}(\text{kgm})$	-1.93	$M_{qq}(\text{kgm}^2/\text{rad}^2)$	-188
$M_{uq}(\text{kgm/rad})$	-2	$M_{uw}(\text{kg})$	24
$Z_{\dot{q}}(\text{kgm}^2/\text{rad})$	-1.93	$Z_{qq}(\text{kgm/rad}^2)$	-0.632
$Z_{\dot{w}}(\text{kg})$	-35.5	$Z_{ww}(\text{kg/m})$	-131
$Z_{uw}(\text{kg/m})$	-28.6	$Z_{uq}(\text{kg/rad})$	-5.22

The nonlinear model of the AUV (23a)–(23d) can be linearized through SIMULINK linearization mode at a steady state [5]

$$\begin{aligned} w &= w_0 + w' \\ q &= q_0 + q' \\ z &= z_0 + z' \\ \theta &= \theta_0 + \theta' \end{aligned} \tag{24}$$

where $[w', q', z', \theta']^T$ denote the tiny linearization error and the steady state point is set to $[0, 0, 2.0, 0]^T$. The linearized AUV model is derived directly

$$\begin{aligned} \dot{\boldsymbol{\chi}} &= A\boldsymbol{\chi} + B\boldsymbol{u} + \boldsymbol{\xi} \\ \boldsymbol{y} &= C\boldsymbol{\chi} \end{aligned} \tag{25}$$

where coefficient matrices $A, B,$ and C are given by

$$A = \begin{bmatrix} -1.0421 & 0.7856 & 0 & 0.0207 \\ 6.0038 & -0.6624 & 0 & -0.7083 \\ 1.0000 & 0 & 0 & -2.0000 \\ 0 & 1.0000 & 0 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 0.0153 & 0.0035 \\ -0.0035 & 0.1209 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

and the output $\boldsymbol{y} = [z, \theta]^T$.

Since the state and output variables are all measurable, an LQI controller is designed to solve the depth control problem for the linearized AUV model shown in Fig. 8. A feedback controller is designed as

$$\boldsymbol{u} = K[\boldsymbol{\chi}, \boldsymbol{\epsilon}]^T = K_{\boldsymbol{\chi}}\boldsymbol{\chi} + K_{\boldsymbol{\epsilon}}\boldsymbol{\epsilon} \tag{27}$$

where $\boldsymbol{\epsilon}$ is the output of the integrator

$$\boldsymbol{\epsilon}(t) = \int_0^t (\boldsymbol{y}_r - \boldsymbol{y})dt. \tag{28}$$

The gain matrix K is obtained by solving an algebraic Riccati equation, which is derived from the minimization of the following cost function:

$$J(\boldsymbol{u}) = \int_0^t \left\{ \begin{bmatrix} \boldsymbol{\chi} & \boldsymbol{\epsilon} \end{bmatrix} Q \begin{bmatrix} \boldsymbol{\chi} \\ \boldsymbol{\epsilon} \end{bmatrix} + \boldsymbol{u}^T R\boldsymbol{u} \right\} dt. \tag{29}$$

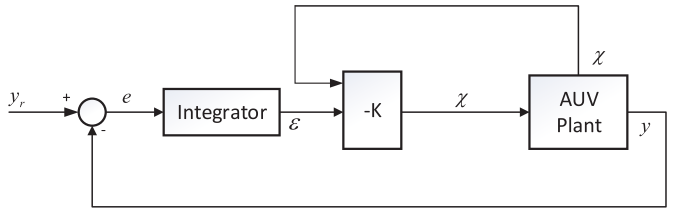


Fig. 8. Structure diagram of LQI.

C. Nonlinear Model Predictive Control

The LQI controller is designed based on an linearized AUV model which is an approximation of the original nonlinear model. We adopt a nonlinear controller derived from nonlinear MPC (NMPC) under an exact nonlinear AUV dynamics. NMPC designs a N-steps cumulative cost function [15]

$$J_k = \frac{1}{2}\boldsymbol{x}_{k+N}^T P_0 \boldsymbol{x}_{k+N} + \frac{1}{2}\sum_{i=0}^{N-1}\left(\boldsymbol{x}_{k+i}^T Q \boldsymbol{x}_{k+i} + \boldsymbol{u}_{k+i}^T R \boldsymbol{u}_{k+i}\right). \tag{30}$$

For each time step k, NMPC predicts an optimal N-step control sequence $\{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+N-1}\}$ by minimizing the optimization function

$$\begin{aligned} &\underset{\{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+N-1}\}}{\text{minimize}} \quad J_k \\ &\text{s.t.} \ \ \boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i, \ \boldsymbol{u}_i) \ \ i = k, \ldots, k+N-1 \end{aligned} \tag{31}$$

where the count of predicting steps N is also called prediction horizon. NMPC solves the N-step optimization problem by iterating two processes alternatively. The forward process executes the system equation using a candidate control sequence to find the predictive state sequence $\{\boldsymbol{x}_{k+i}\}$. The backward process finds the Lagrange multipliers to eliminate the partial derivative terms of J_k with respect to the state sequence, and then updates the control sequence along the gradient vector. The two processes are repeated until the desired accuracy.

D. Experimental Settings

In this section, we introduce the experiment settings for the simulations. The LQI and NMPC controllers are implemented on the MATLAB R2017a platform using control system and MPC toolboxes. As mentioned, the AUV model is linearized by the SIMULINK from the S-function of the AUV dynamics (23a)–(23d). The NNDPG is implemented by Python 2.7 on Linux system using the Google's open source module Tensorflow.

It should be noticed that all the controllers and models are implemented as discrete time versions with sample time $dt = 0.1$ s, although some of them are described under continuous time in the previous sections. For example, the general AUV dynamics (1) is discretized using the forward Euler formula

$$\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_k + dt \cdot f\left(\boldsymbol{\chi}_k, \boldsymbol{u}_k\right). \tag{32}$$

The sample horizon is set to $T = 100$ s.

The disturbance term $\boldsymbol{\xi}$ is generated by an Ornstein–Uhlenbeck process [29]

$$\boldsymbol{\xi}_{k+1} = \beta\left(\boldsymbol{\mu} - \boldsymbol{\xi}_k\right) + \sigma\boldsymbol{\varepsilon} \tag{33}$$

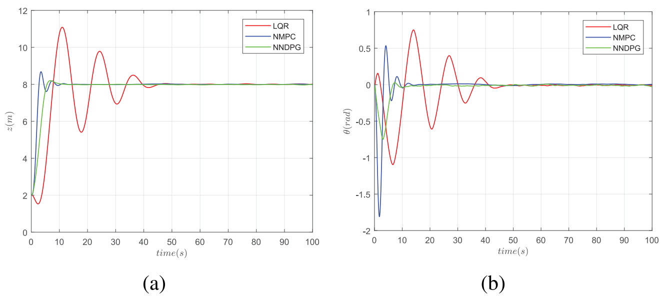


Fig. 9. Tracking behaviors of the three controllers for the constant depth control problem, where $z_r = 8.0$ m and $\theta_r = 0.0$ rad. Tracking trajectories of the (a) depth z and (b) pitch angle θ.

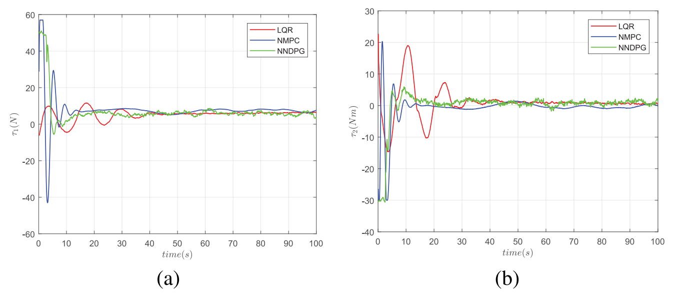


Fig. 10. Control sequences of the three controllers for the constant depth control problem. Trajectories of (a) τ_1 and (b) τ_2.

TABLE II
PERFORMANCE INDEXES OF THREE CONTROLLERS FOR THE CONSTANT DEPTH CONTROL

Controllers	LQI	NMPC	NNDPG
SSE_z(m)	0.0436	0.0094	0.0191
OS_z(m)	3.0849	0.6772	**0.1945**
RT_z(s)	42.5	7.8	**7.3**
SSE_θ(rad)	0.0158	0.0065	0.0108
RT_θ(rad)	46.5	12.2	**11.0**

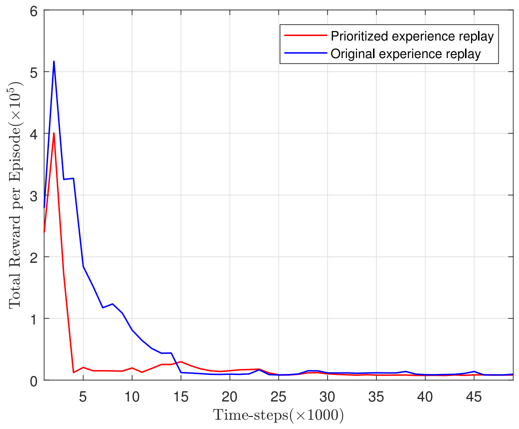


Fig. 11. Comparison of prioritized experience replay and experience replay.

where $\boldsymbol{\varepsilon}$ is a noise term complying standard normal distribution, and other parameters are set as $\boldsymbol{\mu} = \mathbf{0}$, $\beta = 0.15$, and $\sigma = 0.3$. Note that this is a temporal correlated random process.

E. Simulation Results for Constant Depth Control

First, we compare the performance of NNDPG with those of LQI and NMPC on the constant depth control problem. Fig. 9 shows the tracking behaviors of the three controllers from an initial depth $z_0 = 2.0$ m to a target depth $z_r = 8.0$ m. We use three indices, steady-state error (SSE), overshooting (OS), and response time (RT), to evaluate the performance of a controller, whose exact values are present in Table II.

We can find that the LQI behaves worst among the controllers. This result illustrates that the performance of the model-based controller will deteriorate under an inexact model.

In addition, the simulation shows that the performance of NNDPG is comparable with that of NMPC based on a perfect nonlinear AUV model, and even defeats the latter with a faster convergence speed and a smaller OS (bold decimals in Table II). It illustrates the effectiveness of the proposed algorithm under an unknown AUV model.

Fig. 10 shows the control sequences of the three controllers. The control policy learned by NNDPG changes more sensitively than the other controllers. We explain the phenomenon by the approximation error of the neural network. LQI and NMPC can obtain smoother control law functions since they can access the dynamical equations of the AUV. However, a neural network (policy network) is used to generate the control sequence in NNDPG. Therefore, it can be regarded as a compensation for the unknown dynamical model.

To validate the improved data-efficiency of the prioritized experience replay, we compare its performance with the original experience replay through the converging process of the total reward, shown in Fig. 11. We find that the NNDPG with prioritized experience replay spends less steps in converging than the one with original experience replay, since the former replays previous experiences by a more efficient way.

F. Simulation Results for Curved Depth Control

In this section, the AUV is controlled to track a curved depth trajectory. We set the tracked trajectory as a sinusoidal function $z_r = z_{r0} - \sin(\pi/50 \cdot x)$, where $z_{r0} = 10$ m. At first, we assume that the NNDPG has the trend information about the trajectory including the slope angle and its derivative as the situation of the curved depth control studied in Section III. Then we validate the algorithm under the situation where the slope angle is not measurable. Instead, a preceding historical sequence of the measured relative depths $[\Delta z_{t-N}, \Delta z_{t-N+1}, \ldots, \Delta z_{t-1}, \Delta z_t]$ is offered where the length of the sequence is called *window size*. The tracking errors and trajectories are shown in Fig. 12(a) and (b), where NNDPG-PI denotes the NNDGP algorithm with the information about the slope angle, and NNDPG-WIN-X denotes the NNDPG involved with X recently measured relative depths.

First, we observe from Fig. 12 that the performance of the NNDPG-PI is comparable with that of the NMPC, which is similar with the case of the constant depth control. Note that the NMPC needs the accurate dynamics of the AUV

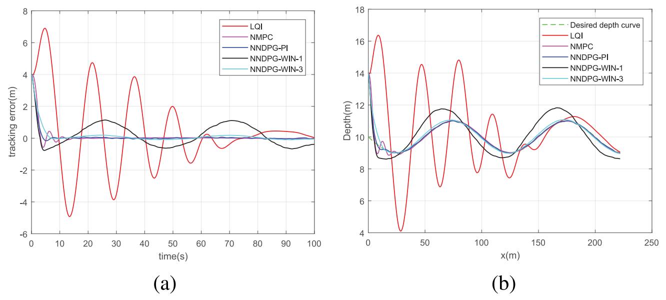


Fig. 12. Tracking errors and trajectories of the curved depth control for LQI, NMPC, and different versions of NNDPG.

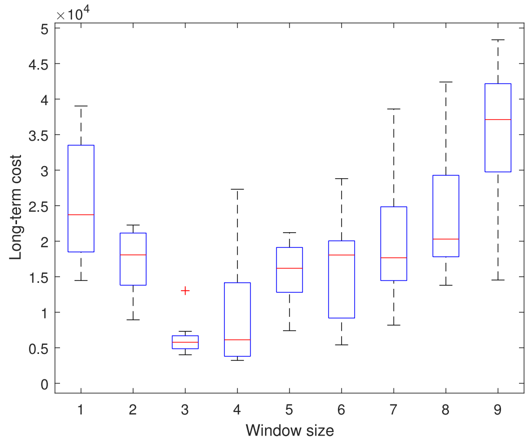


Fig. 13. Boxplot of long-term cost for different window size.

while our algorithm is totally model-free. In addition, the performance of NNDPG-PI is the best one among the different NNDPG versions, while NNDPG-WIN-1 only including the current relative depth performs much worse. It validates that the state designed for the constant depth control problem becomes partially observable for the curved depth setting. However, when we add the recent measured relative depths to the state (NNDPG-WIN-3), the tracking error is much reduced. The improvement can be explained by the implicit trend information containing in the latest measurements.

To determine the best choice of the window size, we list and compare the performances of NNDPG with different window sizes from 1 to 9. The performances are evaluated according to the long-term cost of one experiment

$$J = \sum_{k=1}^{T} \gamma^{k-1} c(\boldsymbol{s}_k, \boldsymbol{u}_k). \quad (34)$$

Because there are disturbances existing in the AUV dynamics, we make ten experiments for each window size. The results are presented in Fig. 13 as a boxplot to show the distributions of the performances.

It is clear to see that the supplement of the past measured relative depths does compensate the missing trend information of the desired depth trajectory. However, it does not mean more is better. Actually, the records beyond too many steps before are shown to deteriorate the performance because they may disturb the learned policy. From the plot we find the best value of the window size is 3, which corresponds to the minimal mean and variance.

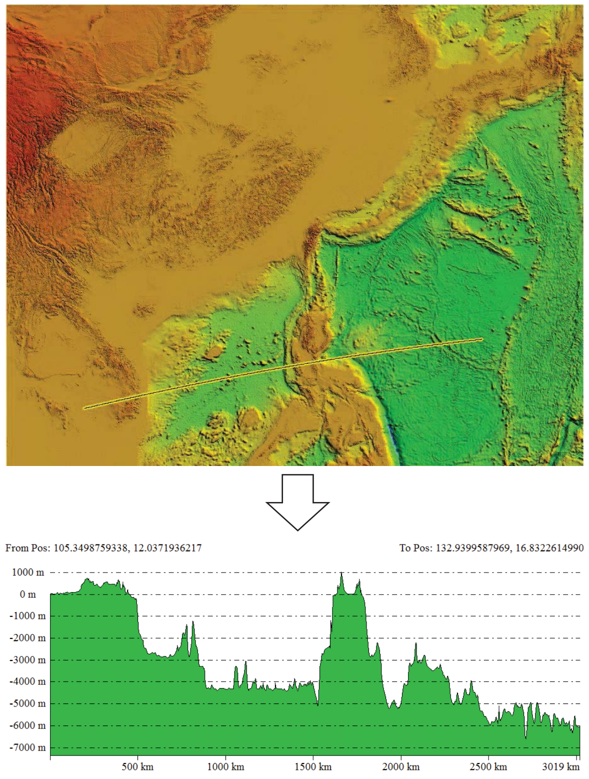


Fig. 14. Process of sampling the depths data along a preset path.

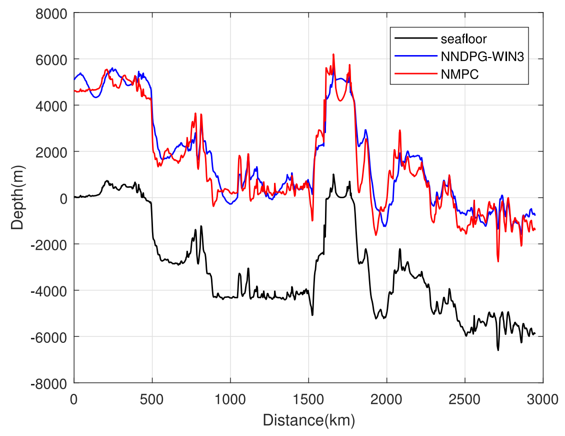


Fig. 15. Tracking trajectory of NNDPG-WIN-3, NMPC, and the realistic seafloor.

G. Simulation Results of Realistic Seafloor Tracking

In the end, we test the proposed RL framework for tracking a real seafloor. The data set sampled from the seafloor of the South China Sea at (23°06′N, 120°07′E) is provided by the Shenyang Institute of Automation, Chinese Academy of Science. We sample the depths along a preset path and obtain a 2-D distance-depth seafloor curve shown in Fig. 14.

Our objective is to control the AUV to track the seafloor curve and keep a constant safe distance simultaneously. We compare the performance of the NNDPG-WIN-3 with that of the NMPC shown in Fig. 15. It seems that both controllers can track the rapidly changing trend of the seafloor pretty well, but our algorithm does not require the dynamics of the AUV.

VII. Conclusion

This paper has proposed a model-free RL framework for the depth control problems of AUVs in discrete time. Three different depth control problems were studied and modeled as MDPs with well-designed states and one-step cost functions. An RL algorithm NNDPG was proposed to learn a state-feedback controller represented by a neural network called the policy network. The weight of the policy network was updated by an approximate policy gradient calculated by the DPG theorem, while another evaluation network was used to estimate the state-action value function and updated by the TD algorithm. The alternative updates of two networks composed of one iteration step of NNDPG. To improve the convergence, the prioritized experience replay was proposed to replay previous experiences to train the network.

We tested the proposed model-free RL framework on a classical REMUS AUV model and compared the performance with those of two model-based controllers. The results showed that the performance of the policy found by NNDPG can compete with those of the controllers under the exact dynamics of the AUV. In addition, we found that the observability of the chosen state influenced the performance and the recent history could be added to improve the performance.

In the future, we will verify the proposed model-free RL framework on a real underwater vehicle which is a deep-sea controllable visual sampler operating at 6000 m under the sea level.

References

[1] A. Kenny *et al.*, "An overview of seabed-mapping technologies in the context of marine habitat classification," *ICES J. Marine Sci. J. du Conseil*, vol. 60, no. 2, pp. 411–418, 2003.

[2] J. A. Farrell, S. Pang, and W. Li, "Chemical plume tracing via an autonomous underwater vehicle," *IEEE J. Ocean. Eng.*, vol. 30, no. 2, pp. 428–442, Apr. 2005.

[3] Z. Liu, P. Smith, T. Park, A. A. Trindade, and Q. Hui, "Automated contaminant source localization in spatio-temporal fields: A response surface and experimental design approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 569–583, Mar. 2017.

[4] K.-P. Lindegaard, "Acceleration feedback in dynamic positioning," Ph.D. dissertation, Dept. Eng. Cybern., Norwegian Univ. Sci. Technol., Trondheim, Norway, 2003.

[5] M. H. Khodayari and S. Balochian, "Modeling and control of autonomous underwater vehicle (AUV) in heading and depth attitude via self-adaptive fuzzy PID controller," *J. Marine Sci. Technol.*, vol. 20, no. 3, pp. 559–578, 2015.

[6] H.-J. Wang, Z.-Y. Chen, H.-M. Jia, and X.-H. Chen, "NN-backstepping for diving control of an underactuated AUV," in *Proc. Oceans*, Waikoloa, HI, USA, 2011, pp. 1–6.

[7] D. Zhu, Y. Zhao, and M. Yan, "A bio-inspired neurodynamics based backstepping path-following control of an AUV with ocean current," *Int. J. Robot. Autom.*, vol. 27, no. 27, pp. 298–307, 2012.

[8] M.-C. Fang and J.-H. Luo, "On the track keeping and roll reduction of the ship in random waves using different sliding mode controllers," *Ocean Eng.*, vol. 34, nos. 3–4, pp. 479–488, 2007.

[9] S. Wen, M. Z. Q. Chen, Z. Zeng, X. Yu, and T. Huang, "Fuzzy control for uncertain vehicle active suspension systems via dynamic sliding-mode approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 1, pp. 24–32, Jan. 2017.

[10] X. Ai, K. You, and S. Song, "A source-seeking strategy for an autonomous underwater vehicle via on-line field estimation," in *Proc. Int. Conf. Control Autom. Robot. Vis.*, Phuket, Thailand, 2016, pp. 1–6.

[11] Z. Li *et al.*, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 6, pp. 740–749, Jun. 2016.

[12] D. Maalouf, A. Chemori, and V. Creuze, "L1 adaptive depth and pitch control of an underwater vehicle with real-time experiments," *Ocean Eng.*, vol. 98, pp. 66–77, Apr. 2015.

[13] T. I. Fossen, A. Loría, and A. Teel, "A theorem for UGAS and ULES of (passive) nonautonomous systems: Robust control of mechanical systems and ships," *Int. J. Robust Nonlin. Control*, vol. 11, no. 2, pp. 95–108, 2001.

[14] A. Budiyono, "Model predictive control for autonomous underwater vehicle," *Indian J. Geo Marine Sci.*, vol. 40, no. 2, pp. 191–199, 2010.

[15] G. J. Sutton and R. R. Bitmead, "Performance and computational implementation of nonlinear model predictive control on a submarine," in *Nonlinear Model Predictive Control*. Basel, Switzerland: Birkhäuser, 2000, pp. 461–472.

[16] E. Campos, A. Chemori, V. Creuze, J. Torres, and R. Lozano, "Saturation based nonlinear depth and yaw control of underwater vehicles with stability analysis and real-time experiments," *Mechatronics*, vol. 45, pp. 49–59, Aug. 2017.

[17] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1141–1153, Sep. 2013.

[18] P. Rakshit *et al.*, "Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multirobot path planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 814–831, Jul. 2013.

[19] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Auton. Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[20] C. Zhou and Q. Meng, "Dynamic balance of a biped robot using fuzzy reinforcement learning agents," *Fuzzy Sets Syst.*, vol. 134, no. 1, pp. 169–187, 2003.

[21] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1–8.

[22] M. Wiering and M. V. Otterlo, *Reinforcement Learning: State-of-the-Art*. Heidelberg, Germany: Springer, 2012.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. vol. 1. Cambridge, MA, USA: MIT Press, 1998.

[24] D. Silver *et al.*, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

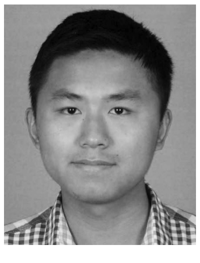
[25] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016.

[26] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, 1992.

[27] H. Pan and M. Xin, "Depth control of autonomous underwater vehicles using indirect robust control method," in *Proc. Amer. Control Conf.*, Montreal, QC, Canada, 2012, pp. 98–113.

[28] P. C. Young and J. C. Willems, "An approach to the linear multivariable servomechanism problem," *Int. J. Control*, vol. 15, no. 5, pp. 961–979, 1972.

[29] P. Mazur, "On the theory of Brownian motion," *Physica*, vol. 25, nos. 1–6, pp. 149–162, 1959.



Hui Wu received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree in control science and engineering with the Department of Automation, Institute of System Integration in Tsinghua University.

His current research interests include reinforcement learning and robot control, especially in continuous control for underwater vehicles.

Shiji Song (SM'17) received the Ph.D. degree in mathematics from the Department of Mathematics, Harbin institute of Technology, Harbin, China, in 1996.

He is currently a Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored over 180 research papers. His current research interests include system modeling, optimization and control, computational intelligence, and pattern recognition.

Keyou You (SM'17) received the B.S. degree in statistical science from Sun Yat-sen University, Guangzhou, China, in 2007, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, in 2012.

He was a Research Fellow with NTU before joining Tsinghua University, Beijing, China, where he is currently an Associate Professor with the Department of Automation. He held visiting positions at the Politecnico di Torino, Turin, Italy, the Hong Kong University of Science and Technology, Hong Kong, and the University of Melbourne, Parkville, VIC, Australia. His current research interests include networked control systems, distributed algorithms, and their applications.

Dr. You was a recipient of the Guan Zhaozhi Award at the 29th Chinese Control Conference in 2010, the CSC-IBM China Faculty Award in 2014, and the National Science Fund for Excellent Young Scholars in 2017. He was nominated for the National 1000-Youth Talent Program of China in 2014.

Cheng Wu received the M.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 1966.

He is a Professor with the Department of Automation, Tsinghua University. He has also been an Academician of the China Engineering Academy since 1995. His current research interests include computer integrated manufacturing systems, manufacturing systems scheduling, optimization of supply chains, and system identification.

Prof. Wu was a recipient of the National Science and Technology Progress Award of China in 1995, 1999, and 2006, and the First Scientific Development Award from the State Educational Committee of China in 1994.