Position Control of an Underwater Biomimetic Vehicle-Manipulator System via Reinforcement Learning

Ruichen Ma[1,2], Yu Wang*[2], Zisen Gao[3], Tianzi Zhao[3], Rui Wang[2], Shuo Wang[1,2], Chao Zhou[4]

1. University of Chinese Academy of Sciences, Beijing 100049, China

2. State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

3. Beijing Institute of Petrochemical Technology, Beijing 102617, China

4. Naval Research Academy, Beijing 100072, China

Abstract: This paper addresses a position control method of an underwater biomimetic vehicle-manipulator system (UBVMS) through reinforcement learning. The system description of the UBVMS with undulating fins is given. Considering the force/torqu generated by undulating fins and hydrodynamic force/torqu, the dynamic model of this UBVMS is established. The position control problem is modeled into a continuous-state, continuous-action Markov decision process (MDP) with a deterministic state transition algorithm based on the dynamic model. To solve this MDP, a reinforcement learning method is presented, which is based on the deep deterministic policy gradient (DDPG) theorem. The simulations of the position control in 5 cases are shown in the end.

Key Words: Position Control, Underwater Biomimetic Vehicle-manipulator System (UBVMS), Deep Deterministic Policy Gradient (DDPG), Reinforcement Learning

1 Introduction

Underwater vehicle-manipulator systems (UVMSs) have the maneuverability, controllability of underwater vehicles and working capacity of various operation manipulators. They have shown potentials in underwater tasks like valve turning[1] and autonomous grasping [2].

Most of the UVMSs use screw propellers for thrust production, but this propulsive method may cause underwater noise and effect the environment and fishes[3]. The relationship curve between thrust and rotating speed of a screw propeller is nonlinear[4], and a certain amount of rotating speed is required to produce stable thrust. So it's difficult for a screw propeller to produce a small amount of thrust when the vehicle is required to implement a small movement.

Propulsion of ribbon undulating fin can produce stable thrust and has a nearly liner thrust production curve[5]. This propulsion imitates the swimming of the South American knifefish [6] and the thrust is generated by a undulating ribbon fin using sinusoidal-based wave patterns. Many biomimetic underwater vehicles (BUVs) with undulating fins are developed. A knifefish robot was built in Northwestern University, it had a ribbon fin with 32 fin rays [7]. A robotic manta ray with two flapping pectoral fins [8] was built in Nanyang Technological University.

In our laboratory, instead of using traditional screw propellers, undulating fins were attached to UVMSs. Underwater biomimetic vehicle-manipulator systems (UBVMSs) were design and implemented [9]. Variety control methods were adapted on the robots to complete certain tasks, such as a prediction-based seabed terrain following control [10] and a floating autonomous manipulation control[11].

This work was supported in part by the National Natural Science Foundation of China under Grant U1713222, Grant 61703401, Grant U1806204; and in part by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61421004, in part by Equipment Pre-research Field Fund under Grant 61403120407, in part by Youth Innovation Promotion Association CAS. (Corresponding author: Yu Wang, yu.wang@ia.ac.cn)

Nowadays many control theories are proposed , based on precise mathematical models or can apply adaptive methods to control a system with uncertainties[12]. In this paper, a reinforcement learning method based on deep deterministic policy gradient (DDPG) theorem is presented. With this method, a control policy has been trained for position control of an UBVMS.

The remainder of this paper is organized as follows: In Section 2, the system description of the UBVMS with undulating fins is introduced. In Section 3, a dynamic model of the UBVMS is established. In Section 4, the position control problem is formed into a MDP with designs of a continuous state, a continuous action, an instant cost and a deterministic state transition algorithm based on the dynamic model. In Section 5, a reinforcement learning method based on DDPG theorem is presented to solve the MDP. In Section 6, the training result of the reinforcement learning algorithm is presented. The position control is implemented in 5 cases and the simulation results are analyzed.

2 System Description

The position control is implemented on an UBVMS. Some relevant equipments and materials are introduced in this section.

2.1 System Configuration of the UBVMS

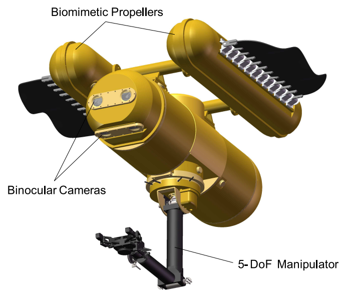

Fig. 1: The design of the UBVMS

The prototype design of the UBVMS is shown in fig. 1. It possess two biomimetic propellers to achieve multimodal motions, a 5-DoF (degree of freedom) manipulator for underwater operation and two pairs of binocular cameras for underwater detection and observation. Some actual measured parameters of the UBVMS are listed in the Table 1.

Table 1: Measured parameters of the UBVMS

Parameters	Values	Parameters	Values
Width	1.21m	Mass	120.6kg
Length	1.20m	Buoyancy	1183.20N
Height	0.95m	Maximum speed	0.2 m/s

2.2 The Kinematics of the Undulating Fins

The biominetic underwater propeller possesses a flexible undulating fin which consists of twelve short rays connected by a black silicone sheet. By distributing all rays in interval phases of a sinusoid, propulsive force can be generated by the propeller. As shown in Fig. 2, the motion model of a single fin ray is:

$$\theta(s,t) = \Theta \sin 2\pi(\frac{s}{\lambda} + ft + \phi) \tag{1}$$

where $\theta(s,t)$ defines the angular deflection of the ray at distance s along the axis X of coordinate system O-XYZ at time t. Θ is the maximum angular deflection of the sinusoidal waves. λ is the wavelength. f is the frequency of the waves. $2\pi\phi$ is the initial phase of the waves.

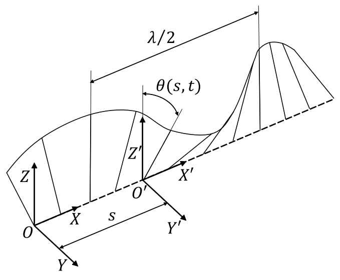

Fig. 2: Sinusoidal wave pattern

When value of f is not 0, with the time t goes, a wave is considered propagating along the axis X. The force generated by this wave can be divided into two types: thrust T along the axis X and lateral force L along the axis Z. The force along the axis Y has little effect and is ignored.

3 Dynamic Model of the UBVMS

The training environment for the RL algorithm is based on the dynamic model of the UBVMS, here three DoF of kinematics and dynamics are considered. The coordinate system is shown in Fig. 3. To describe the position and posture change of the UBVMS, a fixed coordinate system E-xy and a moving coordinate systerm O-uv are established. The origin of system O-uv is fixed on the geometric centre of the robot.

In the system E-xy, the pose of the robot is represented by a vector:

$$\boldsymbol{\chi} = [x, y, \psi]^{\mathrm{T}} \tag{2}$$

where x and y denote displacements of the robot on the two axes. ψ denotes angular displacement of the robot on horizontal plane.

In the system O-uv, the velocity of the robot is determined by a vector:

$$\boldsymbol{v} = [u, v, r]^{\mathrm{T}} \tag{3}$$

where u and v denote linear velocity of the robot. r denotes angular velocity of the robot.

Therefore, the coordinate transformation matrix (from O-uv to E-xy) can be calculated by:

$$\boldsymbol{J} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{4}$$

The force and torque exerted on the UBVMS is expressed as a vector:

$$\boldsymbol{\tau} = [X, Y, N]^{\mathrm{T}} \tag{5}$$

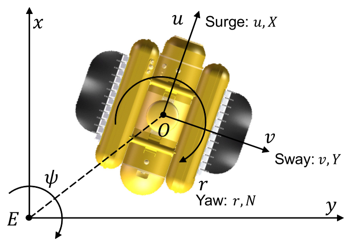

Fig. 3: The coordinate system of the UVMS

Based on the momentum theorem, the dynamic equation of the UBVMS can be described as:

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{v})\boldsymbol{v} = \boldsymbol{\tau} \tag{6}$$

where $\boldsymbol{M}$ represents the generalized mass matrix of the robot, $\boldsymbol{C}(\boldsymbol{v})$ represents the matrix of Coriolis and centripetal force:

$$\boldsymbol{M} = \begin{bmatrix} m & 0 & -my_G \\ 0 & m & mx_G \\ -my_G & mx_G & I_z \end{bmatrix} \tag{7}$$

$$\boldsymbol{C}(\boldsymbol{v}) = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix} \tag{8}$$

where m represents the mass of the UBVMS, x_G, y_G represent the barycentric coordinates. I_z represents the moment of inertia of the robot on the horizontal plane. The software SolidWorks is used to calculate these parameters. A three-dimensional model of the UBVMS has been built in the software, by setting material properties to all parts of the robot's model, the parameters can be obtained: $x_G = 0.015m$, $y_G = 0.008m$, $I_z = 15.44kg \cdot m^2$, here m is set a actual measured value as $m = 120.6kg$.

The force and torque exerted on the UBVMS can be divided into three parts:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_p - \boldsymbol{\tau}_h - \boldsymbol{\tau}_d \tag{9}$$

where $\boldsymbol{\tau}_p$ is the force and torque generated by the biomimetic propellers, $\boldsymbol{\tau}_h$ is the force and torque describing the hydrodynamic effect, $\boldsymbol{\tau}_d$ denotes other disturbances like the disturbing force caused by cables on the robot. In this paper, $\boldsymbol{\tau}_d$ is ignored.

$$\boldsymbol{\tau}_h\left(\boldsymbol{v},\dot{\boldsymbol{v}}\right)=\begin{bmatrix} X_{uu}u^2+X_{vv}v^2+X_{rr}r^2+X_{vr}vr+X_{\dot{u}}\dot{u} \\ Y_{uu}u^2+Y_{uv}uv+Y_{ur}ur+Y_{v|v|}v|v|+Y_{r|r|}r|r|+Y_{v|r|}v|r|+Y_{\dot{v}}\dot{v}+Y_{\dot{r}}\dot{r} \\ N_{uu}u^2+N_{uv}uv+N_{ur}ur+N_{v|v|}v|v|+N_{r|r|}r|r|+N_{\dot{v}}\dot{v}+N_{\dot{r}}\dot{r} \end{bmatrix} \tag{10}$$

3.1 Hydrodynamic Force/Torque

The Hydrodynamic model of a underwater vehicle can be described with 108 coefficients[13]. Considering only the motions on the horizontal plane, the hydrodynamic force can be simplified as Eq. 10

Calculated by a computational fluid dynamics (CFD) simulation software Fluent, the hydrodynamic coefficients in Eq. 10 are listed in Table 2.

Table 2: Hydrodynamic coefficients of the UBVMS

X_{uu}	0.0290	Y_{uu}	-0.1790	$Y_{v\|r\|}$	-17.6802	N_{ur}	0.0388
X_{vv}	1.0647	Y_{uv}	-0.4818	$Y_{\dot{v}}$	-0.1896	$N_{v\|v\|}$	-2.8041
X_{rr}	-0.7551	Y_{ur}	-0.9651	$Y_{\dot{r}}$	0.0362	$N_{r\|r\|}$	-2.5915
X_{vr}	-2.7321	$Y_{v\|v\|}$	-14.5121	N_{uu}	0.0280	$N_{\dot{v}}$	-0.0344
$X_{\dot{u}}$	-0.0568	$Y_{r\|r\|}$	-0.0412	N_{uv}	-0.6180	$N_{\dot{r}}$	-0.0152

3.2 Propeller Force/Torque

The relationship between the generated force of the propeller and the frequency of wave is tested. Other variables of wave shown in Eq. 2 are set at certain values which are tested in a previous study[14]. Resulting data of thrust T and lateral force L are obtained from a measurement platform built in a previous study[14]. The experimental data and the fitting cure of the two force are shown in Fig. 4.

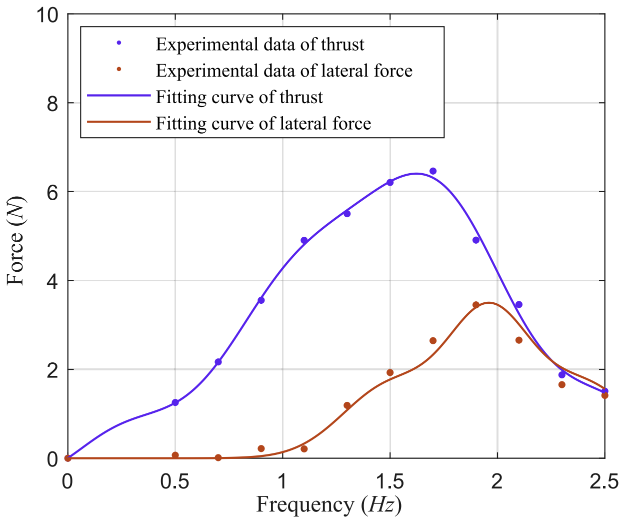

Fig. 4: Thrust and lateral force as functions of frequency.

The software Matlab is used for curve fitting and the fitted formulas and related parameters calculated from Matlab are shown below.

Fitted formula of thrust:

$$T\left(f\right)=\sum_{n=1}^{3}a_n e^{-\left(\frac{f-b_n}{c_n}\right)^2} \tag{11}$$

Fitted formula of lateral force:

$$L\left(f\right)=q_0+\sum_{n=1}^{4}\left(q_n\cos\left(nwf\right)+p_n\sin\left(nwf\right)\right) \tag{12}$$

Table 3: Fitted parameters of formula of thrust

n	1	2	3
a_n	431.5	-597.8	169.8
b_n	1.96	1.96	1.96
c_n	0.3573	0.3669	0.3947

Table 4: Fitted parameters of formula of lateral force

n	1	2	3	4
q_n	-3.018	0.4614	-0.1803	0.03193
p_n	0.6315	-0.09298	0.02593	0.2848
other	$q_0=2.709,\ w=1.957$			

The vector $\boldsymbol{\tau}_p$ representing the force and torque exerted on the UBVMS by the two biomimetic propellers can be expressed as:

$$\boldsymbol{\tau}_p(\boldsymbol{f})=\begin{bmatrix} T(f_1)+T(f_2) \\ L(f_1)-L(f_2) \\ D_1T(f_1)-D_2T(f_2) \end{bmatrix} \tag{13}$$

where $\boldsymbol{f}=[f1,f2]^{\mathrm{T}}$ is the input frequency of the biomimetic propellers, $D_i(i=1,2)$ indicates the distance from the force application point of the propeller to the origin of the moving coordinate system: $D_1=D_2=0.495m$.

4 Markov Decision Process Modeling

Reinforcement learning is a dynamic-programming-based method to solve Markov decision process (MDP). Before applying the reinforcement learning method to solve the problem of position control, we need to transform the problem into a MDP.

4.1 MDP for the Position Control

A MDP is a stochastic process, which satisfies the Markov property. A MDP includes four parts: 1) a state space $\mathcal{S}$; 2) an action space $\mathcal{A}$; 3) a cost function of one step $c\left(\boldsymbol{s},\boldsymbol{a}\right)$: $\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$; 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 currant state only depends on the last state and action.

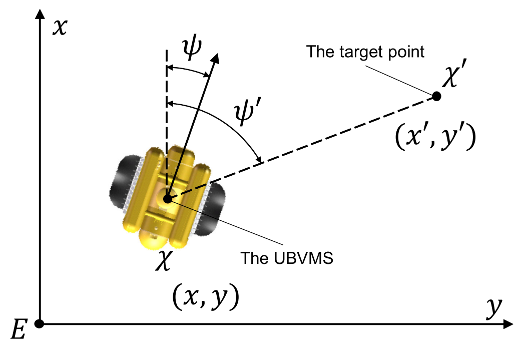

Fig. 5: The coordinate system for MDP modeling

The state of position control is designed as:

$$\boldsymbol{s}=\left[\Delta\boldsymbol{\chi},\boldsymbol{v},\dot{\boldsymbol{v}}\right]^{\mathrm{T}} \tag{14}$$

where $\Delta\boldsymbol{\chi} = \boldsymbol{\chi}' - \boldsymbol{\chi}$ and $\boldsymbol{\chi}' = [x', y', \psi']^{\mathrm{T}}$ is the description of the target point as shown in Fig. 5. x' and y' are coordinates of the target point in the system E-xy. ψ' is the angular displacement form x axis to a connecting line between the robot's geometric centre and the target point.

The action is the control inputs of the biomimetic propellers, therefore:

$$\boldsymbol{a} = \boldsymbol{f} \tag{15}$$

The purpose of the position control is to control the UBVMS move to and stop at any target point from a random start location and random posture, therefore the cost function in this MDP is designed as:

$$c(\boldsymbol{\chi}, \boldsymbol{v}) = \rho_1(x'-x)^2 + \rho_2(y'-y)^2 + \rho_3(\psi'-\psi)^2 + \rho_4 k(\boldsymbol{v}) \tag{16}$$

where $\rho_i (i = 1, 2, 3, 4)$ is the weight of each item, $\rho_1(x' - x)^2$ and $\rho_2(y' - y)^2$ are used for minimizing the distance between the robot and the target point. $\rho_3(\psi' - \psi)^2$ guides the robot to turn to and aim at the target point. $k(\boldsymbol{v})$ represents the kinetic energy of the robot, defined as:

$$k(\boldsymbol{v}) = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \tag{17}$$

therefore $\rho_4 k(\boldsymbol{v})$ is used for minimizing the consumed energy.

4.2 Deterministic State Transition

Instead of the transition probability, we use a deterministic state transition algorithm to directly generate the state $\boldsymbol{s}_t$ based on previous state $\boldsymbol{s}_{t-1}$ and action $\boldsymbol{a}_{t-1}$. A detailed procedure of this responding process is shown in Algorithm 1 which is based on the dynamic model of the UBVMS introduced before.

Algorithm 1 Deterministic State Transition

```
Input:
    Previous state $\boldsymbol{s}_{t-1} = [\Delta\boldsymbol{\chi}_{t-1}, \boldsymbol{v}_{t-1}, \dot{\boldsymbol{v}}_{t-1}]^{\mathrm{T}}$ ,previous action $\boldsymbol{a}_{t-1}$, initial location and posture $\boldsymbol{\chi}_{t-1} = [x_{t-1}, y_{t-1}, \psi_{t-1}]^{\mathrm{T}}$, target point information $\boldsymbol{\chi}'$, time interval $dt$.
Output:
    Current state $\boldsymbol{s}_t = [\Delta\boldsymbol{\chi}_t, \boldsymbol{v}_t, \dot{\boldsymbol{v}}_t]^{\mathrm{T}}$
1: Compute the total force and torque exerted on the UBVMS according to Eq. 6 and Eq. 9:
    $\boldsymbol{\tau}_t = \boldsymbol{\tau}_p(\boldsymbol{a}_{t-1}) - \boldsymbol{\tau}_h(\boldsymbol{v}_{t-1}, \dot{\boldsymbol{v}}_{t-1}) - \boldsymbol{C}(\boldsymbol{v}_{t-1})\boldsymbol{v}_{t-1}$
2: Compute the accelerated velocity according to Eq. 6:
    $\dot{\boldsymbol{v}}_t = \boldsymbol{M}^{-1}\boldsymbol{\tau}_t$
3: Compute the velocity: $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \dot{\boldsymbol{v}}_t dt$
4: Compute the coordinate transformation matrix $\boldsymbol{J}$ according to Eq. 4.
5: Compute the displacement:
    $[dx, dy]^{\mathrm{T}} = \boldsymbol{J}[u_t dt, v_t dt]^{\mathrm{T}}$
6: Update the location: $\begin{cases} x_t = x_{t-1} + dx \\ y_t = y_{t-1} + dy \end{cases}$
7: Compute the posture: $\psi_t = \psi_{t-1} + r_t dt$, therefore $\boldsymbol{\chi}_t = [x_t, y_t, \psi_t]^{\mathrm{T}}$ is obtained
8: Compute the relative location and posture:$\Delta\boldsymbol{\chi}_t = \boldsymbol{\chi}' - \boldsymbol{\chi}_t$, therefore all the elements in $\boldsymbol{s}_t$ are acquired.
```

5 Markov Decision Process Solving

A reinforcement learning algorithm based on deep deterministic policy gradient(DDPG) for the MDP solving is proposed in this section.

5.1 Action-Value Function

The performance of a policy in reinforcement learning is often evaluated by action-value function (also called the Q-value function), which is based defined by:

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = E\left[\sum_{i=t}^{T} \gamma^{i-t} c(\boldsymbol{s}_i, \boldsymbol{a}_i) | \boldsymbol{s}_t, \boldsymbol{a}_t\right] \tag{18}$$

which is a long-term cost function at time step t with state s_t and action a_t, under a specific policy π.

The action-value function satisfies the Bellman optimality principle and the policy is considered deterministic in this paper. Therefore, the action-value function can be changed into:

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = E[c(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma Q^{\pi}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}) | \boldsymbol{s}_t, \boldsymbol{a}_t] \tag{19}$$

A commonly used greedy policy $\mu(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} Q(\boldsymbol{s}, \boldsymbol{a})$ can be adapted and the function approximation parameter is set as $\boldsymbol{\beta}$. Therefore the policy can be optimized by minimizing the loss:

$$l_t = y_t - Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t | \boldsymbol{\beta}) \tag{20}$$

where

$$y_t = c(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma Q^{\pi}(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1}) | \boldsymbol{\beta}) \tag{21}$$

To update the parameter $\boldsymbol{\beta}$, the temporal difference (TD) method is applied [15]:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \alpha_{\boldsymbol{\beta}} l_t \nabla_{\boldsymbol{\beta}} Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t | \boldsymbol{\beta}) \tag{22}$$

where $\alpha_{\boldsymbol{\beta}}$ is the updating rate.

5.2 Deterministic Policy Gradient

The control frequencies of the undulating fins is continuous, but the action-value function cannot be straightly applied to continuous action spaces. Instead, an actor-critic based method deterministic policy gradient (DPG) algorithm is used for improving the policy. The DPG algorithm assumes a deterministic parameterized policy function $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ and updates $\boldsymbol{\theta}$ along the negative gradient of the long-term cost function:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t - \alpha_{\boldsymbol{\theta}} \widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} \tag{23}$$

where $\alpha_{\boldsymbol{\theta}}$ is the updating rate, $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})}$ is a stochastic approximation of the true gradient. Combined with the approximated action-value function $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t | \boldsymbol{\beta})$ in the last section of policy $\pi = \mu$ and time step $t = i$. The DPG algorithm is defined as:

$$\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \mu(\boldsymbol{s}_i | \boldsymbol{\theta}) \nabla_{\boldsymbol{a}_i} Q^{\mu}(\boldsymbol{s}_i, \boldsymbol{a}_i | \boldsymbol{\beta}) \tag{24}$$

5.3 Neural Network Approximators

Critic network $Q(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{\beta})$ and actor network $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ are constructed, with $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ as weights.The critic network is designed of four layers with state $\boldsymbol{s}$ inputted in the first layer and action $\boldsymbol{a}$ inputted in the second layer. The output layer is a linear unit to generate a scalar Q-value.The actor network is designed of three layers, with the state $\boldsymbol{s}$ as a input. The output layer uses tanh function to generate the action $\boldsymbol{a}$ with a given range $[-2.5, 2.5]$ according to the control frequency range of the biomimetic propeller. ReLu function is applied to all the hidden layers in both actor and critic networks.

5.4 Reinforcement Learning Algorithm

The reinforcement learning algorithm for the position control is based on deep deterministic policy gradient (DDPG)[16], which is an actor-critic, model-free algorithm and works well over continuous action spaces. A detailed procedure of the algorithm is shown in Algorithm 2. The related parameters used in the algorithm are shown in Table 5.

Algorithm 2 Reinforcement Learning Algorithm

```
Input:
    number of training episode $M$, number of steps of each episode
    $T$, batch size $N$, reward discount $\gamma$, updating rate of the critic
    network $\alpha_\beta$, updating rate of the actor network $\alpha_\theta$, replacing
    rate of the target networks $\tau$, capacity of the replay buffer $R$.
Output:
    Position control policy $\boldsymbol{a} = \mu(\boldsymbol{s}|\boldsymbol{\theta})$
 1: Randomly initialize critic network $Q(\boldsymbol{s}, \boldsymbol{a}|\boldsymbol{\beta})$ and actor network
    $\mu(\boldsymbol{s}|\boldsymbol{\theta})$ with weights $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, Initialize target network
    $Q'$ and $\mu'$ with wights $\boldsymbol{\beta}' \to \boldsymbol{\beta}$ and $\boldsymbol{\theta}' \to \boldsymbol{\theta}$, initialize reply
    buffer with capacity of $R$.
 2: for episode = 1 to M do
 3:     Randomly generate a initial state $\boldsymbol{s}_0$
 4:     for t = 0 to T do
 5:         Generate an action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t|\boldsymbol{\theta}) + \mathcal{N}_t$ according to current policy and exploration noise $\mathcal{N}_t$.
 6:         Execute action $\boldsymbol{a}_t$, compute cost $c_t$ according to Eq. 16,
            compute new state $\boldsymbol{s}_{t+1}$ according to Algorithm 1.
 7:         Store transition tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, c_t, \boldsymbol{s}_{t+1})$ in the replay buffer.
 8:         Sample a minibatch of $N$ transitions $(\boldsymbol{s}_i, \boldsymbol{a}_i, c_i, \boldsymbol{s}_{i+1})$
            $(i = 1, \dots, N)$ from $R$.
 9:         Compute $y_i = c_i + \gamma Q'(\boldsymbol{s}_{i+1}, \mu'(\boldsymbol{s}_{i+1}|\boldsymbol{\theta}')|\boldsymbol{\beta}')$,
            $(i = 1, \dots, N)$,
            therefore $l_i = y_i - Q^\pi(\boldsymbol{s}_i, \boldsymbol{a}_i|\boldsymbol{\beta}')$, $(i = 1, \dots, N)$.
10:         Update the critic network:
                $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha_\beta \frac{1}{N}\sum_{i=1}^{N} l_i \nabla_{\boldsymbol{\beta}} Q^\pi(\boldsymbol{s}_i, \boldsymbol{a}_i|\boldsymbol{\beta})$
11:         Compute $\nabla_{\boldsymbol{a}_i} Q^\mu(\boldsymbol{s}_i, \boldsymbol{a}_i|\boldsymbol{\beta})$, $(i = 1, \dots, N)$.
12:         Update the actor network:
                $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}}\mu(\boldsymbol{s}_i|\boldsymbol{\theta})\nabla_{\boldsymbol{a}_i} Q^\mu(\boldsymbol{s}_i, \boldsymbol{a}_i|\boldsymbol{\beta})$
13:         Update the target networks: $\begin{cases}\beta' \leftarrow \tau\beta + (1-\tau)\beta' \\ \theta' \leftarrow \tau\theta + (1-\tau)\theta'\end{cases}$
14:     end for
15: end for
```

Table 5: Input parameters in Algorithm 2

M	T	N	γ	α_β	α_θ	τ	R
2000	500	32	0.9	0.001	0.001	0.0	30000

6 Simulations and Analysis

The reinforcement learning algorithm is established in software PyCharm based on the programming language Python. The networks are generated by the Tensorflow module. The training of the networks and the simulations are all executed in PyCharm and the results are introduced in this section.

6.1 Simulations of the Position Control

As shown in Fig. 6, the simulations consist of 5 cases and all begin with the robot at position and posture $\boldsymbol{\chi}_0 = \left[0, 0, \frac{\pi}{2}\right]^{\mathrm{T}}$ (black dot) in system E-xy, with $\boldsymbol{v}_0 = [0, 0, 0]^{\mathrm{T}}$ and $\dot{\boldsymbol{v}}_0 = [0, 0, 0]^{\mathrm{T}}$. During the simulation, the robot is controlled to move to and stop at a target point following the trained policy. The target points in each case are: (1) $\boldsymbol{\chi}_1' = \left[0, 2, \frac{\pi}{2}\right]^{\mathrm{T}}$ (blue dot), (2) $\boldsymbol{\chi}_2' = \left[2, 2, \frac{\pi}{4}\right]^{\mathrm{T}}$ (orange dot), (3) $\boldsymbol{\chi}_3' = [2, 0, 0]^{\mathrm{T}}$ (yellow dot), (4) $\boldsymbol{\chi}_4' = \left[2, -2, -\frac{\pi}{4}\right]^{\mathrm{T}}$ (purple dot) and (5) $\boldsymbol{\chi}_5' = \left[0, -2, -\frac{\pi}{2}\right]^{\mathrm{T}}$ (green dot). The lines in Fig.6, Fig.7, Fig.9 and Fig.8 use the same color as the dots in Fig.6 to represent the same case of a simulation.

The trajectories of the robot in each simulation are shown in Fig. 6. Each simulation ends when the robot reaches the target within distance of 0.1m for 50 steps. The time interval dt in Algorithm 1 is set $dt = 0.05s$ during each simulation. The control frequencies of the two undulating fins during each simulation are shown in Fig. 7. The instant costs of the MDP in each simulation are shown in Fig. 8. The kinetic energy of the robot in each simulation are shown in Fig. 9.

6.2 Analysis of the Simulation Results

As shown in Fig. 6, the UBVMS can reach the target in all 5 simulations, but only the trajectory towards $\boldsymbol{\chi}_1'$ (blue line) is straight to the target. In other cases, the policy selects to move by a spiral path when the robot is near to the targets. When moving toward $\boldsymbol{\chi}_4'$ (purple line) and $\boldsymbol{\chi}_5'$ (green line), the robot first drives backward, then turns around to aim at the target and complete the remaining path.

The action $\boldsymbol{a}$ consists of the control frequencies of the undulating fins and as shown in Fig. 7, the frequencies are selected almost within $-1.6Hz$ to $1.6Hz$. That's may because the thrust has positive correlation with the control frequency when the value of frequency is under $1.6Hz$ as shown in Fig. 4.

The Instant costs $c(\boldsymbol{\chi}, \boldsymbol{v})$ in Eq. 16 of every step are shown in Fig. 8, the results indicate that the trained policy is reliable to control the robot to reduce instant costs and complete the position control.

The Kinetic energy $k(\boldsymbol{v})$ in Eq. 17 represents the consumed energy and also shows the moving tendency of the robot. Therefore Fig. 9 reveals the speed of the robot in simulations: in all 5 cases, the robot starts with zero speed and can all stop near target in the end, the troughs in purple line and green line also show the turning around processes when moving towards $\boldsymbol{\chi}_4'$ and $\boldsymbol{\chi}_5'$.

7 Conclusion

In this paper, a position control method of an UBVMS is presented. A dynamic model of the UBVMS is established. The position control problem is formed into a MDP with the designs of a continuous state, a continuous action, an instant cost and a deterministic state transition algorithm. A reinforcement learning algorithm based on DDPG theorem is presented to solve the MDP. The position control is implemented in 5 cases. The simulation results are analyzed and validity of the proposed control method is verified.

In the future, we will adapt the proposed position control method on the UBVMS and implement some related exper-

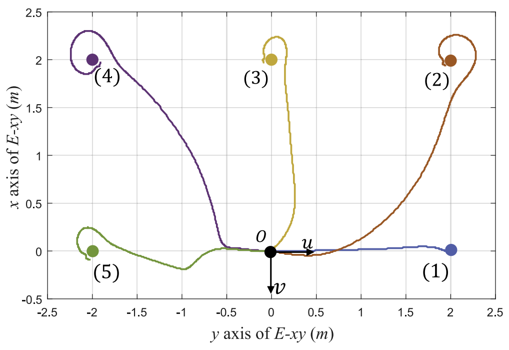

Fig. 6: Trajectories of the UBVMS during simulations

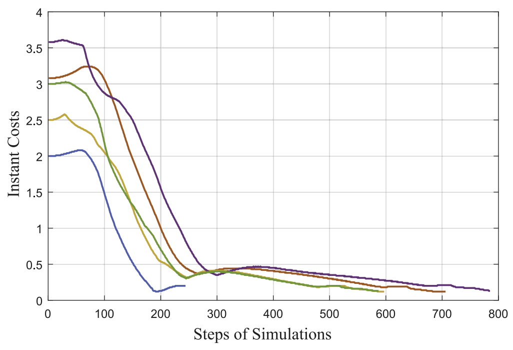

Fig. 8: Instant costs during simulations

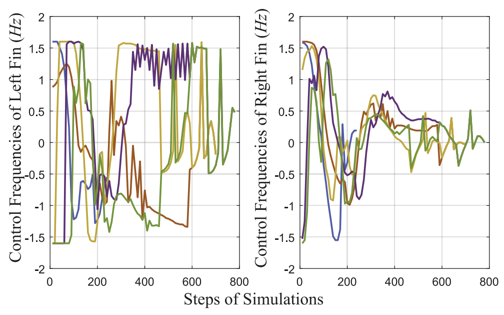

Fig. 7: Control frequencies during simulations

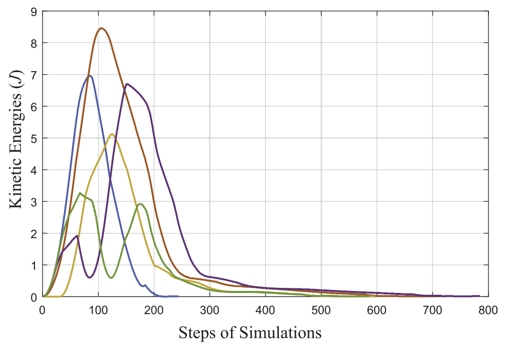

Fig. 9: Kinetic energies during simulations

iments in real underwater environment.

References

[1] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by girona500 uvms: A practical approach to autonomous underwater manipulation," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 529–536.

[2] Q. Zhang, A. Zhang, P. Gong, W. Quan, *et al.*, "Research on autonomous grasping of an uvms with model-known object based on monocular visual system," in *The Twentieth International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 2010.

[3] R. B. Mitson and H. P. Knudsen, "Causes and effects of underwater noise on fish abundance estimation," *Aquatic Living Resources*, vol. 16, no. 3, pp. 255–263, 2003.

[4] Z. Li, J. Tao, Y. Luo, L. Ding, and Z. Deng, "Dynamic analysis of a cable underwater robot in a nuclear reaction pool," in *2016 IEEE International Conference on Mechatronics and Automation*. IEEE, 2016, pp. 2278–2283.

[5] S. Sefati, I. Neveln, M. A. MacIver, E. S. Fortune, and N. J. Cowan, "Counter-propagating waves enhance maneuverability and stability: a bio-inspired strategy for robotic ribbon-fin propulsion," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE, 2012, pp. 1620–1625.

[6] I. D. Neveln, Y. Bai, J. B. Snyder, J. R. Solberg, O. M. Curet, K. M. Lynch, and M. A. MacIver, "Biomimetic and bio-inspired robotics in electric fish research," *Journal of experimental Biology*, vol. 216, no. 13, pp. 2501–2514, 2013.

[7] C. Zhou and K. Low, "Design and locomotion control of a biomimetic underwater vehicle with fin propulsion," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 25–35, 2011.

[8] M. M. Rahman, S. Sugimori, H. Miki, R. Yamamoto, Y. Sanada, and Y. Toda, "Braking performance of a biomimetic squid-like underwater robot," *Journal of Bionic Engineering*, vol. 10, no. 3, pp. 265–273, 2013.

[9] Y. Wang, R. Wang, S. Wang, M. Tan, and J. Yu, "Underwater bio-inspired propulsion: from inspection to manipulation," *IEEE Transactions on Industrial Electronics*, DOI:10.1109/TIE.2019.2944082, 2019.

[10] M. Cai, Y. Wang, S. Wang, R. Wang, L. Cheng, and M. Tan, "Prediction-based seabed terrain following control for an underwater vehicle-manipulator system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2019.2944651, 2019.

[11] C. Tang, Y. Wang, S. Wang, R. Wang, and M. Tan, "Floating autonomous manipulation of the underwater biomimetic vehicle-manipulator system: Methodology and verification," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 4861–4870, 2017.

[12] L. Liu, Y. J. Liu, S. C. Tong, and C. L. P. Chen, "Integral barrier lyapunov function based adaptive control for switched nonlinear systems," *SCIENCE CHINA Information Sciences*, DOI: 10.1007/s11432-019-2714-7, 2019.

[13] H. Li, L. Zhao, and Y. Mao, "Analysis of six-degree-of-freedom motion in submarines under sea disturbance," *Journal of Harbin Engineering University*, no. 1, p. 16, 2017.

[14] R. Ma, H. Du, R. Wang, Y. Wang, S. Wang, C. Tang, *et al.*, "Data-driven locomotive strategies of the uvms propelled by undulating fins," in *The 29th International Ocean and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 2019, pp. 1568–1574.

[15] H. Wu, S. Song, K. You, and C. Wu, "Depth control of model-free auvs via reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

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