

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

Ruichen Ma^{1,2}, Yu Wang^{*2}, Rui Wang², Shuo Wang^{1,2},

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

Abstract: This paper proposes an omnidirectional drift control on an underwater biomimetic vehicle-manipulator system (UBVMS). The UBVMS has two biomimetic propellers, they obtain propulsive force by actuating their undulating fins. A configuration of the system, the dynamics of the UBVMS and the kinematics of the fins are given respectively. Then the control problem is designed as a Markov decision process (MDP). A reinforcement learning method based on the twin delayed deep deterministic policy gradient (TD3) is proposed for this MDP. A control policy is well trained by the reinforcement learning method and tested in eight different simulations. An analysis of the simulation results is also given.

Key Words: Omnidirectional Drift Control, Undulating Fin, Underwater Biomimetic Vehicle-manipulator System (UBVMS), Reinforcement Learning, Twin Delayed Deep Deterministic policy gradient (TD3)

1 Introduction

Exploration of the ocean is a popular topic in recent year. Complex environment is a major challenge for underwater vehicles. The vehicles may face narrow spaces surrounded by reef and exuberant sea-plants may enwind their screw propellers. However aquatic animal swimmers are well adapted to this environment and are skilled at moving through complex spaces[1]. Therefore, many biomimetic underwater vehicles (BUVs) are designed and built to imitate underwater creatures[2].

As for exploitation of the ocean, the underwater vehicle-manipulator system (UVMS) is proposed [3]. A UVMS is usually equipped with a robotic arm with a grasping hand or other equipments for certain tasks.

In our laboratory, we combined the low-speed maneuverability of BUV and the operating ability of UVMS and came up with the underwater biomimetic vehicle-manipulator system (UBVMS)[4] and have implemented many experiments to explore its potentials[5–8].

In this paper, an omnidirectional drift control method for the UBVMS is proposed. Omnidirectional drift means that the robot travels to all directions among horizontal plane while holding its yaw angle still. This control method contains the use of counter-propagating waves[9] generated from undulating fins to gain the driving force for lateral movement. As the control system of undulating fins is strongly coupling, we use a reinforcement learning method based on the twin delayed deep deterministic policy gradient (TD3)[10] to train a control policy for this omnidirectional drift control.

This paper is organized as follows: In section 2, the configuration of the system, the dynamics of the UBVMS and

the kinematics of the undulating fins are introduced. In section 3, the omnidirectional drift control problem is designed and formed into a Markov decision process (MDP). In section 4, a reinforcement learning method based on TD3 is proposed to solve the MDP, and the control policy of omnidirectional drift is well trained. In section 5, the policy is applied to control the UBVMS in eight simulations to test its performance, and the results are analyzed.

2 System Description

In this section, the configuration of the UBVMS, the dynamic model of the UBVMS and the kinematics of the undulating fins are introduced.

2.1 Configuration Overview

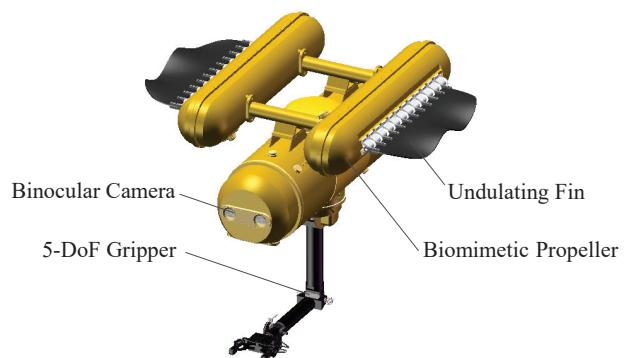


Fig. 1: The configuration of the UBVMS

The UBVMS is driven by two biomimetic propellers, consists of two binocular cameras and a 5-DoF (degree of freedom) gripper. Some related parameters are listed in Table 1.

Table 1: Parameters of the UBVMS

Parameters	Values	Parameters	Values
Width	1210mm	Buoyancy	1183.20N
Length	1200mm	Fin Length	550mm
Height	950mm	Fin Width	175mm
Mass	120.6kg	Fin Thickness	4mm

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFC1512202, in part by the Youth Innovation Promotion Association CAS under Grant 2018162, in part by the National Natural Science Foundation of China under Grant U1713222, Grant 62073316, Grant U1806204, 62033013, in part by key projects of foreign cooperation of CAS under Grant 173211KYSB20200020, in part by the Strategic Priority Research Program of Chinese Academy of Science, Grant No. XDB32050100 (Corresponding author: Yu Wang, yu.wang@ia.ac.cn).

2.2 Dynamics of the UBVMS

The whole system is carried out and analyzed in a simulation environment, therefore the dynamic model of the robot is required. The model is based on a previously built model in [11], but establishes different coordinate systems and uses more control variables to drive the undulating fins.

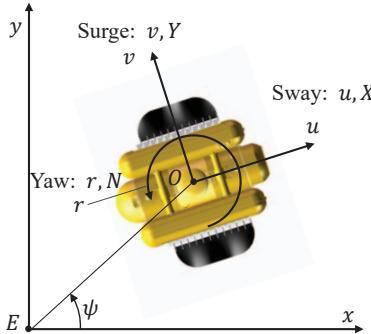


Fig. 2: The coordinate systems

The coordinate systems $E\text{-}xy$ and $O\text{-}uv$ are shown in Fig. 2. $E\text{-}xy$ represents the world coordinate system. $O\text{-}uv$ is the inertial coordinate system of the robot, the origin of $O\text{-}uv$ is the geometric center of the robot. The pose of the robot in $E\text{-}xy$ can be defined as a vector:

$$\chi = [x, y, \psi]^T \quad (1)$$

The velocity of the robot in $O\text{-}uv$ is defined as:

$$v = [u, v, r]^T \quad (2)$$

The force and torque exerted on the robot is:

$$\tau = [X, Y, N]^T \quad (3)$$

The dynamic equation of the UBVMS can be described as:

$$M\dot{v} + C(v)v = \tau \quad (4)$$

where M is the generalized mass matrix of the robot, $C(v)$ is the matrix of Coriolis and centripetal force. The description of M and $C(v)$ are introduced in [11]. The force and torque τ consists of three parts: the driving power of propellers, the hydrodynamic effect and the disturbances, they are all introduced in [11].

2.3 Kinematics of the Undulating Fins

The inward counter-propagating sinusoidal waves described in [12] is used as the basic wave pattern to apply on the undulating fins. At actual usage, the wavelength λ is half of the fin length l , initial phase $2\pi\phi$ is considered as zero and the maximum angular deflection Θ is set as $\frac{\pi}{6}$. Thus the kinematics of the pattern used in this paper is defined as:

$$\begin{cases} \theta_1(s, t) = \frac{\pi}{6} \sin 2\pi \left(\frac{2s}{l} + ft \right), & s \leq d \\ \theta_2(s, t) = \frac{\pi}{6} \sin 2\pi \left(4\eta - \frac{2s}{l} - ft \right), & s > d \end{cases} \quad (5)$$

$\theta_i(s, t)$ ($i = 1, 2$) is the angular deflection of a fin ray at distance s in time t . f is the frequency of wave. d is the propagating distance of the first wave. η is the position ratio, defined as $\eta = \frac{d}{l}$.

3 Markov Decision Process Modeling

A MDP consists of four parts: a state space, an action space, a one-step cost function and an one-step transition probability. The omnidirectional drift control problem should be designed as a MDP, so that the reinforcement learning methods can be applied.

3.1 Description of the Omnidirectional Drift Control

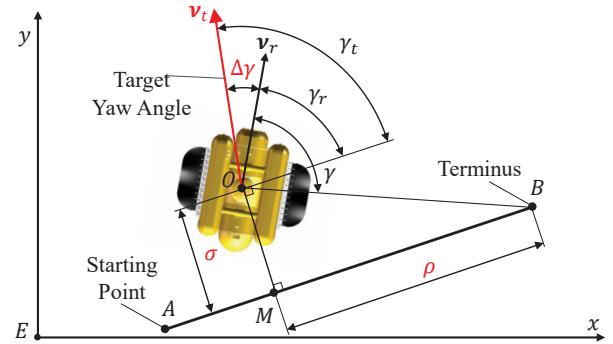


Fig. 3: The coordinate systems for drift control

As shown in Fig. 3, AB is the trajectory line tracked by the robot from starting point A to terminus B , M is a point in AB that is the nearest to the robot. ρ is the remaining distance which is the length of line MB . σ is the tracking error which is the length of line MO .

Vector ν_t is the target yaw angle, vector ν_r is the yaw angle of the robot. γ_t is the angular deflection from ν_t to line AB . γ_r is the angular deflection from ν_r to line AB . γ is the angular deflection from ν_r to line OB . $\Delta\gamma$ is the angular error, defined as: $\Delta\gamma = \gamma_t - \gamma_r$.

To achieve the omnidirectional drift, the robot should be controlled to fulfill three tasks simultaneously: 1) Reach the terminus, therefore reduce the remaining distance ρ to zero. 2) Minimize the tracking error σ . 3) Minimize the absolute value of the angular error $\Delta\gamma$.

3.2 MDP for the Omnidirectional Drift Control

The whole control process is modeled as a MDP:

First, the state of the omnidirectional drift control is designed as:

$$s = [\rho, \sigma, \gamma_t, \gamma_r, \gamma, v, \xi]^T \quad (6)$$

all these state parameters are necessary, they describe how the robot perceives the environment. ξ is the error indicator, defined as:

$$\xi = \begin{cases} 1, & \sigma < \sigma_{safe} \text{ and } |\Delta\gamma| < \gamma_{safe} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where σ_{safe} and γ_{safe} are desired error ranges, thus ξ shows whether the errors are acceptable.

Second, the action of the drift control is based on the control variables in Eq. 5:

$$a = [\eta_1^*, f_1^*, \eta_2^*, f_2^*]^T \quad (8)$$

where subscripts 1 and 2 indicate left and right fins, the asterisk superscript means the variable is uniformed its range to $[-1, 1]$.

Third, the cost function is used to lead the robot to accomplish the task, therefore is designed as:

$$c = p_1\rho + p_2\sigma + p_3|\Delta\gamma| + p_4k(\mathbf{v}) - p_5\xi \quad (9)$$

where $p_i (i = 1, \dots, 5)$ is the weight of each part. $k(\mathbf{v})$ is the kinetic energy, defined as: $k(\mathbf{v}) = \mathbf{v}^T M \mathbf{v}$. It's noteworthy that $-p_5\xi$ is subtractive, it is used as an extra reward if the robot keeps itself in the desired error range.

Last, as for the transition probability, a deterministic state transition algorithm is designed to simulate this process[11].

4 Markov Decision Process Solving

To solve the MDP, a reinforcement learning algorithm based on the twin delayed deep deterministic policy gradient [10] method is proposed.

4.1 Background

Q-value function is often used to evaluate the performance of a policy in reinforcement learning processes, which is defined as:

$$Q^\pi(s_t, a_t) = E \left[\sum_{i=t}^T \gamma^{i-t} c(s_i, a_i) | s_t, a_t \right] \quad (10)$$

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

The Q-value function satisfies the Bellman optimality principle, therefore Eq. 10 is reformed as:

$$Q^\pi(s_t, a_t) = E [c(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t, a_t] \quad (11)$$

A generally used greedy policy $\mu(s) = \arg \max_a Q(s, a)$ is applied in reinforcement learning algorithm. It can be optimized by minimizing the loss:

$$l_t = y_t - Q^\pi(s_t, a_t | \beta) \quad (12)$$

where β is the approximation parameter and

$$y_t = c(s_t, a_t) + \gamma Q^\pi(s_{t+1}, \mu(s_{t+1}) | \beta) \quad (13)$$

is the target action-value function.

An iteration method, temporal difference (TD) is applied to update the parameter β :

$$\beta_{t+1} = \beta_t + \alpha_\beta l_t \nabla_\beta Q^\pi(s_t, a_t | \beta) \quad (14)$$

where α_β is the updating rate.

To change the Q-value function to adapt continuous action space, deterministic policy gradient (DPG) is applied. In DPG, a deterministic parameterized policy function $\mu(s|\theta)$ is designed, where θ is the approximation parameter and it can be updated by:

$$\theta_{t+1} \doteq \theta_t - \alpha_\theta \widehat{\nabla_\theta J(\theta)} \quad (15)$$

where α_θ is the rate of updating, the true gradient is approximated by $\widehat{\nabla_\theta J(\theta)}$. Combined with the Q-value function under policy $\pi = \mu$ and time $t = i$, the approximation of the true gradient is defined as:

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N} \sum_{i=1}^N \nabla_\theta \mu(s_i | \theta) \nabla_{a_i} Q^\mu(s_i, a_i | \beta) \quad (16)$$

4.2 Double Q-learning and Target Policy Smoothing

Double Q-Learning method is used by TD3, therefore two target Q-value functions in Eq. 13 are constructed:

$$\begin{cases} y_{1t} = c(s_t, a_t) + \gamma Q_1^\pi(s_{t+1}, \mu(s_{t+1}) | \beta_1) \\ y_{2t} = c(s_t, a_t) + \gamma Q_2^\pi(s_{t+1}, \mu(s_{t+1}) | \beta_2) \end{cases} \quad (17)$$

To avoid over approximation, a new target Q-value function based on Eq. 17 is designed:

$$y_t = c(s_t, a_t) + \gamma \min_{i=1,2} Q_i^\pi(s_{t+1}, \mu(s_{t+1}) | \beta_i) \quad (18)$$

To eliminate the estimated deviation in reinforcement learning. A regularization is added to actors, therefore Eq. 18 is changed into:

$$y_t = c(s_t, a_t) + \gamma \min_{i=1,2} Q_i^\pi(s_{t+1}, \mu(s_{t+1}) | \beta_i + \varepsilon) \quad (19)$$

where $\varepsilon \sim \text{clip}(N(0, \tilde{\sigma}), -c, c)$ is also referred as noise.

4.3 Neural Network Approximators

Two critic networks $Q_1(s, a | \beta_1), Q_2(s, a | \beta_2)$ and an actor network $\mu(s | \theta)$ are designed. Two critic networks share a same structure of four layers: State s is the input of the first layer. Action a is the input of the second layer. A scalar Q-value is generated from the output layer. The actor network has three layers: State s is the input of the first layer. The output layer generate the action a in range $[-1, 1]$ according to Eq. 8 using the Tanh function. All the hidden layers of the three networks use the ReLu function as transfer function.

4.4 Reinforcement Learning Algorithm

A procedure of the algorithm based on TD3 is given in Algorithm 1. The involved parameters are listed in Table 2.

Table 2: Input parameters in Algorithm 1

M	T	N	γ	α_β	α_θ	τ	R	d
20000	500	32	0.99	0.001	0.001	0.995	10000	2

4.5 Training Results of the Algorithm

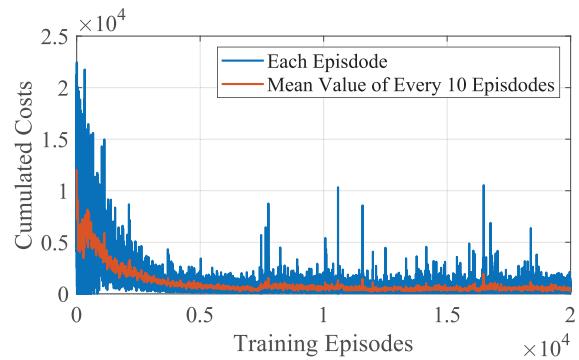


Fig. 4: Cumulated costs of training episodes

The cost function in Eq. 9 shows an instant cost of one simulation step. By adding up T numbers of costs c in one episode, a cumulated cost is acquired. Fig. 4 shows the cumulated costs of all M episodes. The training reaches a convergence around 5000 episodes and then the control policy becomes steady.

Algorithm 1 Reinforcement Learning Algorithm Based on TD3

Input:

Number of episodes M , steps in each episode T , batch size N , reward discount γ , updating rate of the two critic networks α_β , updating rate of the actor network α_θ , replacing rate of the three corresponding target networks τ , capacity of the replay buffer R , update delay steps d .

Output:

Trained control policy $a = \mu(s|\theta)$

- 1: Initialize two critic network $Q_1(s, a|\beta_1), Q_2(s, a|\beta_2)$ and actor network $\mu(s|\theta)$
 - 2: Initialize target network Q'_1, Q'_2 and μ' using wights $\beta'_1 \rightarrow \beta_1, \beta'_2 \rightarrow \beta_2$ and $\theta' \rightarrow \theta$,
 - 3: Initialize the reply buffer with capacity of R .
 - 4: **for** episode = 1 to M **do**
 - 5: Generate an initial state s_0 randomly
 - 6: **for** t = 0 to T **do**
 - 7: Generate an action $a_t = \mu(s_t|\theta) + \varepsilon$ according to Eq. 19.
 - 8: Execute a_t , compute c_t based on Eq. 9, compute s_{t+1} using the deterministic state transition algorithm[11].
 - 9: Store transition tuple (s_t, a_t, c_t, s_{t+1}) in the replay buffer.
 - 10: Sample a minibatch of N transitions (s_n, a_n, c_n, s_{n+1}) ($n = 1, \dots, N$) from R .
 - 11: Compute

$$y_n = c_n + \gamma \min_{i=1,2} Q_i^\pi(s_{n+1}, \mu(s_{n+1})|\beta_i + \varepsilon)$$

$$(n = 1, \dots, N), (i = 1, 2).$$
 therefore $l_n = y_n - Q^\pi(s_n, a_n|\beta'), (n = 1, \dots, N).$
 - 12: Update the critic network:

$$\beta_i \leftarrow \beta_i + \alpha_\beta \frac{1}{N} \sum_{n=1}^N l_n \nabla_\beta Q_i^\pi(s_n, a_n|\beta_i), (i = 1, 2)$$
 - 13: Compute $\nabla_{a_n} Q^\mu(s_n, a_n|\beta), (n = 1, \dots, N).$
 - 14: **if** $t \bmod d$ **then**
 - 15: Update actor network:

$$\theta \leftarrow \theta - \alpha_\theta \frac{1}{N} \sum_{n=1}^N \nabla_\theta \mu(s_n|\theta) \nabla_{a_n} Q_1^\mu(s_n, a_n|\beta_1)$$
 - 16: Update target networks:

$$\beta'_i \leftarrow \tau \beta_i + (1 - \tau) \beta'_i, (i = 1, 2)$$

$$\theta' \leftarrow \tau \theta + (1 - \tau) \theta'$$
 - 17: **end if**
 - 18: **end for**
 - 19: **end for**
-

5 Simulations and Analysis

The trained control policy of omnidirectional drift control is adapted to the dynamic model of the UBVMS and tested in several simulations to evaluate its performance.

5.1 Simulations of the Omnidirectional Drift Control

The simulations are carried out in eight situations, the parameters of each simulation are listed in Table 3. All the simulations of omnidirectional drift control share a same starting point but different terminuses. The UBVMS are controlled by the trained policy and drift from the starting point to each terminus while holding the yaw angle still.

Table 3: Parameters of simulations

No.	Terminus		Starting Point	
	Position	Yaw Angle	Position	Yaw Angle
1	$(0, 2)$	$\frac{\pi}{2}$	$(0, 0)$	$\frac{\pi}{2}$
2	$(\sqrt{2}, \sqrt{2})$			
3	$(2, 0)$			
4	$(\sqrt{2}, -\sqrt{2})$			
5	$(0, -2)$			
6	$(-\sqrt{2}, -\sqrt{2})$			
7	$(-2, 0)$			
8	$(-\sqrt{2}, \sqrt{2})$			

5.2 Analysis of the Omnidirectional Drift Control

The trajectories of the UBVMS in simulations are shown in Fig. 5. Each trajectory is colored differently to distinguish the eight simulations. In all simulations, the tasks can be finished by the robot, but their performances are divergent.

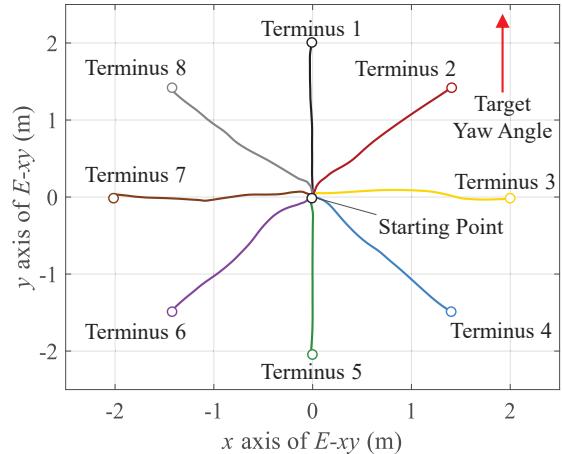


Fig. 5: Trajectories of the UBVMS during simulations

Simulations are evaluated in three aspects: the remaining distance ρ in Fig. 6, the tracking error σ in Fig. 7 and the angular error $\Delta\gamma$ in Fig. 8. The line colors in Fig. 6 ~ Fig. 8 match the colors in Fig. 5. Therefore, the eight simulations can be distinguished according to the eight colors.

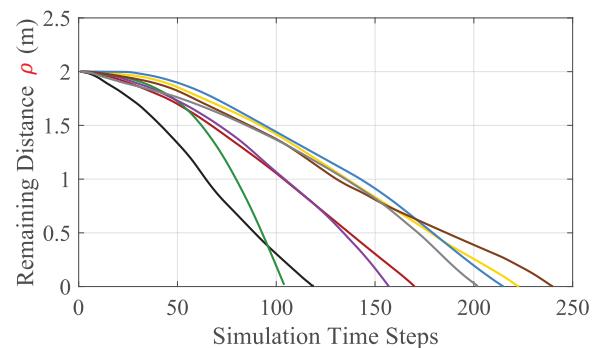


Fig. 6: Remaining distance during simulations

The robot in simulation 1 (black line) travels forward towards terminus and the robot in simulation 2 (green line) travels backward towards terminus. In these two simulations, the robots spend fewer simulation time steps to reach

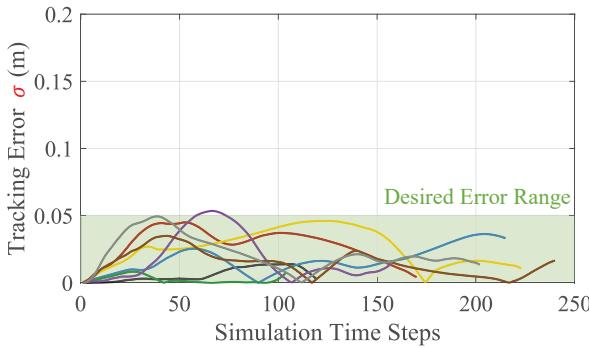


Fig. 7: Tracking error during simulations

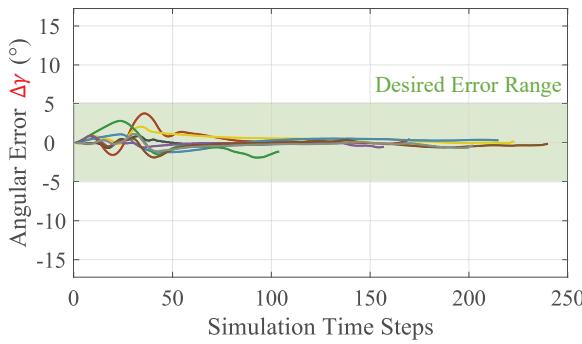


Fig. 8: Angular error during simulations

terminuses and their tracking error σ and angular error $\Delta\gamma$ are also relatively small.

The robots in simulation 3 (yellow line) and simulation 7 (brown line) drift laterally towards terminuses on right and left sides. They spend more time than the robots in other simulations.

The robots in all eight simulations except simulation 6 (purple line) can keep the tracking error γ and the angular error $\Delta\sigma$ within the desired error range defined in Eq. 7 during entire simulations. And in simulation 6, the robot only break through the desired error range around time steps of 55 to 75 as shown in Fig. 8 which is still acceptable.

6 Conclusion

In this paper, an omnidirectional drift control method is proposed. The controlled subject is an UBVMS, which possesses two biomimetic propellers with undulating fins. The configuration of the system, the dynamics of the UBVMS and the kinematics of the fins are introduced. The omnidirectional drift control problem is designed as a MDP. The control policy is acquired by training the MDP using a TD3-based reinforcement learning method. In the end, the trained control policy is implemented on the UBVMS in eight simulations to test its performance and the simulation results are analyzed.

References

- [1] O. M. Curet, N. A. Patankar, G. V. Lauder, and M. A. MacIver, “Mechanical properties of a bio-inspired robotic knifefish with an undulatory propulsor,” *Bioinspiration & biomimetics*, vol. 6, no. 2, p. 026004, 2011.
- [2] R. Wang, S. Wang, Y. Wang, L. Cheng, and M. Tan, “Development and motion control of biomimetic underwater robots: A survey,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020, doi: 10.1109/TSMC.2020.3004862.
- [3] W. Chen, Y. Wei, J. Zeng, X. Jia, and Z. Wang, “Review of underwater vehicle-manipulator system control method,” *Journal of Chongqing University of Technology (Natural Science)*, p. 08, 2015.
- [4] Y. Wang, R. Wang, S. Wang, M. Tan, and J. Yu, “Underwater bioinspired propulsion: from inspection to manipulation,” *IEEE Transactions on Industrial Electronics*, vol. 67, no. 9, pp. 7629–7638, 2019.
- [5] 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*, 2019, doi: 10.1109/TSMC.2019.2944651.
- [6] X. Bai, Y. Wang, S. Wang, R. Wang, M. Tan, and W. Wang, “Modeling and analysis of an underwater biomimetic vehicle-manipulator system,” *SCIENCE CHINA Information Sciences*, doi: 10.1007/s11432-020-3054-7.
- [7] R. Wang, S. Wang, Y. Wang, M. Tan, and J. Yu, “A paradigm for path following control of a ribbon-fin propelled biomimetic underwater vehicle,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 3, pp. 482–493, 2017.
- [8] R. Wang, S. Wang, Y. Wang, M. Cai, and M. Tan, “Vision-based autonomous hovering for the biomimetic underwater robot—robocutt-ii,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8578–8588, 2018.
- [9] 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.
- [10] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.
- [11] R. Ma, Y. Wang, Z. Gao, T. Zhao, R. Wang, S. Wang, and C. Zhou, “Position control of an underwater biomimetic vehicle-manipulator system via reinforcement learning,” in *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE, 2020, pp. 573–578.
- [12] R. Ma, Y. Wang, R. Wang, and S. Wang, “Development of a propeller with undulating fins and its characteristics,” in *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2019, pp. 737–742.