Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Path-following Control of Fish-like Robots: A Deep Reinforcement Learning Approach ⋆

Tianhao Zhang ** **Runyu Tian** ** **Chen Wang** *,** **Guangming Xie** **,***

* *National Engineering Research Center for Software Engineering, Peking University, Beijing 100871, China (e-mail: wangchen@pku.edu.cn)*
** *The State Key Laboratory of Turbulence and Complex Systems, Intelligent Biomimetic Design Lab, College of Engineering, Peking University, Beijing 100871, China (e-mail: {tianhao_z, trytian, xiegming}@pku.edu.cn)*
*** *Institute of Ocean Research, Peking University, Beijing 100871, China*

Abstract: In this paper, we propose a deep reinforcement learning (DRL) approach for path-following control of a fish-like robot. The desired path may be a randomly generated Bézier curve. First, to implement the locomotion control of the fish-like robot, we design a modified Central Pattern Generated (CPG) model, using which the fish achieves varied swimming behaviors just by adjusting a single control input. To reduce the reality gap between simulation and the physical system, using the experimental data of the real fish-like robot, we build a surrogate simulation environment, which also well balances the accuracy and the speed of training. Second, for the path-following control, we select the advantage actor-critic (A2C) approach and train the control policy in the surrogate simulation environment with a straight line as the desired path. Then the trained control policy is directly deployed on a physical fish-like robot to follow a randomly generated Bézier curve. The experimental results show that our proposed approach has good practical applicability in view of its efficiency and feasibility in controlling the physical fish-like robot. This work shows a novel and promising way to control biomimetic underwater robots in the real world.

Keywords: Reinforcement learning control, autonomous underwater vehicles, biomimetic underwater robots, path following, deep learning

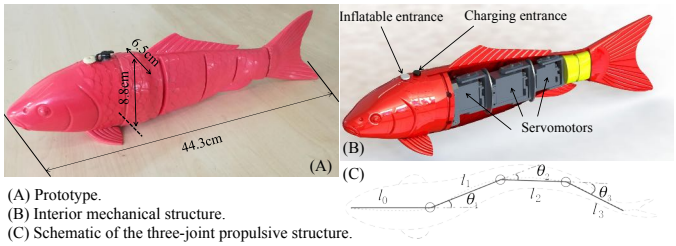
1. INTRODUCTION

Biomimetic robots have attracted increasing attention and significant progress has been made, since such robots replicate the outstanding skills of organisms in nature and, in turn, become powerful tools for understanding animal behavior (Butail et al. (2015)) and helping people in various tasks (Wang et al. (2017)). The fish-like robot is a typical underwater biomimetic robot. A variety of robotic fish prototypes have been constructed, most of which are designed based on the anguilliform swimming mode and the carangiform mode, such as the well-known RoboTuna (Streitlien et al. (1996)), the salamander robot (Ijspeert et al. (2007)), and the carp robot focused in this paper (see Fig. 1). With the superior performance brought by bionics, fish-like robots have been utilized for a growing variety of applications (Ryuh et al. (2015)).

Many applications in mobile robots are built upon the functionality of accurately following a predefined geometric path, which is one of the central problems in automatic guidance (Kapitanyuk et al. (2018)). Various control algorithms have been developed to investigate the path-following task for underwater vehicles, including the proportional-integral-derivative (PID) control (Lekkas and Fossen (2012)), fuzzy control (Zhu et al. (2016)), adaptive control (Shin et al. (2017)) and so on. Unfortunately, most of the algorithms require prior knowledge of dynamic modeling, which is not easy to be obtained for a real underwater vehicle in an uncertain environment. Obviously, the situation will be more complicated for a fish-like robot who has a flexible body, since the robot's interaction with its hydrodynamical environment is uncertainty and variability. To overcome such limitations, recently, Woo et al. (2019) proposed a deep reinforcement learning (DRL) based controller for path following of an unmanned surface vehicle (USV). Their controller does not require any prior knowledge of USV dynamics. However, they only considered the linear path following that the desired path is a straight line. And most important, their work focused on the USV of a rigid body, whose dynamics is much simpler than that of a fish-like robot. To the best of our knowledge, there is no existing work of controlling a physical fish-like robot without knowing its dynamic model to follow a randomly generated Bézier curve as the desired path.

In this paper, a DRL based approach is proposed for path-following control of a fish-like robot in the real world. First, we design a modified Central Pattern Generated (CPG) model to implement the lower level locomotion control of the fish-like robot. Varied swimming behaviors can be achieved by the single control input of the CPG model. Based on the experimental data of the real fish-like robot, we build a surrogate simulation environment, which well balances the accuracy and the speed of training. Second, we choose the advantage actor-critic (A2C) approach and train the control policy in the surrogate simulation environment with a straight line as the desired path. Then the trained policy is directly deployed on a physical fish-like robot to follow a randomly generated Bézier curve. We will show

⋆ This work was supported in part by grants from the National Natural Science Foundation of China (NSFC, No. 61973007, 61633002, U1909206). *(Corresponding author: Chen Wang.)*



(A) Prototype.
(B) Interior mechanical structure.
(C) Schematic of the three-joint propulsive structure.

Fig. 1. The Fish-like robot.

through the experiments that our proposed approach has good practical applicability in view of its efficiency and feasibility in controlling the fish-like robot in the real world.

The main contributions of this paper are twofold. First, as far as the authors are aware, it is the first time that DRL is applied in the control of a biomimetic underwater robot with a flexible body whose accurate dynamic model is almost impossible and unnecessary to know. Second, using our proposed DRL approach in the path-following control of a fish-like robot, the control policies for the robot are **trained only in simulation** and only with a straight line as the desired path, the trained policy is **directly deployed on the real fish-like robot without any tedious tuning** and still performs well with a randomly generated Bézier curve as the desired path. In summary, this work shows a novel and promising way to control the biomimetic underwater robots in the real world.

2. CPG MODEL FOR LOCOMOTION CONTROL

In this section, we first introduce the fish-like robot, and then propose a modified CPG model to control its locomotion.

2.1 Fish-like Robot

In this work, a widely concerned fish-like robot (Wang et al. (2011); Yu et al. (2016)) was chosen as the biomimetic underwater robot of interest. The physical structure of the robotic fish mimics a typical carangiform fish, Koi Carp, which consists of a streamlined head, a flexible body, and a caudal fin (Fig.1). The head of the robotic fish concludes an STM 32 onboard control unit, a wireless module, and four $5V$ batteries. The body of the fish-like robot contains three revolute joints that are linked together by aluminium exoskeletons, and each joint is driven by an R/C servomotor. Each servomotor at each joint is steered by a PWM signal generated from the onboard control unit, which controls its relative joint angle with respect to those of its adjacent joints. The caudal fin of the fish is attached to the third joint. θ_i and l_i $(i = 1, 2, 3)$ are the deflection angle and the joint length of the corresponding joint i, respectively. The fish-like robot measures 44.3cm long and weighs 0.85kg in total. Its density is just a little bit smaller than that of the water so that it swims just below the water surface.

2.2 Robot's Locomotion Control

The CPGs are essential building blocks for the locomotion neural circuits found in both invertebrates and vertebrates (Ijspeert (2008)). A key feature of the CPGs is the capability of producing coordinated patterns of rhythmic activities without any rhythmic inputs from sensory feedback or high-level control signals. Thus the lower level locomotion control of the fish-like robot is implemented based on a simple but effective CPG model (Wang et al. (2011); Li et al. (2015)). Due to the one-to-one correspondence between the oscillators of the CPG and the joints of the robot, for the three-joint fish-like robot of interest, the ith, $i = 1, 2, 3$, oscillator is implemented as follows,

$$\begin{cases} \dot{r}_i(t) = & \zeta_r(R_i - r_i(t)) \\ \dot{x}_i(t) = & \zeta_x(X_i - x_i(t)) \\ \ddot{\phi}_i(t) = & -\zeta_\phi^2 \sum_{j=1,j\neq i}^{3}(\phi_i(t) - \phi_j(t) - \varphi_{ji}) \\ & -4\zeta_\phi(\dot{\phi}_i(t) - 2\pi f) \\ \theta_i(t) = & x_i(t) + r_i(t)\sin(\phi_i(t)) \end{cases} \tag{1}$$

where f represents the desired swing frequency of each joint, R_i and X_i denote the desired swing amplitude and offset angle of the joint i, respectively, and φ_{ij} is set to be the desired phase bias between joint i and j. Three parameters $\zeta_r, \zeta_x, \zeta_\phi$ affect the transient dynamics of the amplitude $r_i(t)$, offset angle $x_i(t)$, and phase $\phi_i(t)$ of each joint i, respectively. The CPG model's output signals $\theta_i(t)$, which represent the deflection angle of the corresponding joint i at time t (Fig. 1(C)), are sent to each servomotor at each joint in the form of PWM to control the robot's locomotion.

Such a CPG model (1) can spontaneously generate rhythmic output signals to propel the robotic fish and easily change its locomotion behavior by adjusting the input parameters. However, the number of the parameters (12 in total) is too much which brings the difficulty of policy training, thus we need a modified CPG model that can generate varied output signals with only a few control inputs. To this end, we first fixed some of the parameters as $f = 1Hz$, $[R_1, R_2, R_3] = [0.0873, 0.1746, 0.2619]rad$, $[\varphi_{12}, \varphi_{23}] = [1.396, 2.094]rad$, $\zeta_r = 11.68/s$, $\zeta_\phi = 5.84/s$ according to the physical characteristics of the fish-like robot, and let the desired offset angles of the three joints to be equal $X_c \triangleq X_i, i = 1, 2, 3$, so that only two parameters X_c and ζ_x were left. Note that, for $X_c = 0$, the fish shifts its tail first to the left (resp. right) and then back to the middle during the first (resp. second) half of each swing period $T = \frac{1}{f} = 1s$. Based on this observation, we consider the situation in discrete time and set X_c by step signals

$$X_c(t) = \begin{cases} u(k), & t \in [kh, kh + \frac{h}{2}) \\ 0, & t \in [kh + \frac{h}{2}, kh + h) \end{cases} \tag{2}$$
$$\forall t \in [kh, kh + h),\ k = 0, 1, 2, \ldots$$

where k is the time step, $h = \frac{T}{2}$ is the sampling period. In this context, $u(k) \in [U_{min}, U_{max}]$, $k = 0, 1, 2, \ldots$ are a series of control inputs under which the offset angle of each joint of the robot can be set during each sampling period $t \in [kh, kh + h)$, where the range $[U_{min}, U_{max}]$, $U_{min} < 0 < U_{max}$ depends on the physical constraints of the robot. Meanwhile, we set the parameter $\zeta_x = 14.4$ to ensure the offset angle $x_i(t)$ of each joint i first increase to the control input $u(k)$ when $t \to kh + \frac{h}{2}$ and then go back to 0 when $t \to kh + h$. It is interesting to note that, the way the control input $u(k)$ works, which divides each swing period T into two halves, allows the offset angle $x_i(t)$ to have more flexible changes. Therefore, to distinguish the two halves of each swing period, we introduced a time tag $t_a(k) \in \{0, 1\}$ that for even $k = 0, 2, 4, \ldots$ and $t_a(k) = 1$ for odd $k = 1, 3, 5, \ldots$; thus $t_a(k) = 0$ (resp. $t_a(k) = 1$) implies that $[kh, kh + h)$ is the first (resp. second) half of each swing period of the robotic fish.

Up to now, we obtained a modified CPG model (1)(2) with only one control input $u(k)$, who can deliver much more kinds of output signals (i.e., swimming behaviors) than that of the original CPG model.

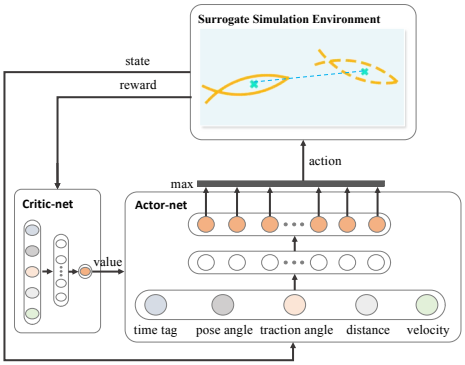


Fig. 2. Structure of the control policy networks.

3. DRL APPROACH FOR PATH-FOLLOWING CONTROL

In this section, we propose a DRL based approach to deal with the path-following control task of the fish-like robot. An overview of our DRL approach is shown in Fig.2, and the training loop proceeds as follows. Initially, within the surrogate simulation environment we designed, it starts a training episode and outputs the states of the fish-like robot. The control policy, implemented by an actor-network, maps the observations of current states to an action which determines the robot's turning motion. Then it executes the action within the surrogate simulation environment, and calculates the corresponding reward and the next states. Subsequently, the policy evaluation is provided by a critic-network to help the actor with the policy improvement. In the direction suggested by the critic, the actor-network updates policy parameters, and the training episode continues until the task is finished or failed. After each training episode, the average reward of some latest episodes decides whether the learning is finished or not. In what follows, we describe each component in details.

3.1 Description of Path-following Task

Path-following task requires the robot to accurately follow a predefined geometric path. Here we use a parametric Bézier curve of degree n to describe the desired path $\mathcal{P}$ as

$$\mathbf{B}(w) = \sum_{i=0}^{n} \mathbf{P}_i \frac{n!}{i!(n-i)!}(1-t)^{n-i}w^i, \ w \in [0,1] \quad (3)$$

where w is a parameter, and the point $\mathbf{P}_i$ are control points. Fig.3 schematically shows the geometry of the path-following task of a fish-like robot, which combines the ideas of the line-of-sight (LOS) guidance (in blue) (Fossen and Pettersen (2014)) and the nonlinear guidance law (NLGL) (in green) (Sujit et al. (2014); Oh et al. (2015)).

In our case, the initial position of the robot is set on the right side of the directed path $\mathcal{P}$. Borrowing the idea of LOS approach, the robot location $\mathbf{p}$ has a unique projection $\mathbf{P}$ onto the path $\mathcal{P}$. At each time step k, the signed distance from the robot location $\mathbf{p}$ to the desired path $\mathcal{P}$ is defined as $d(k) = \text{dist}(\mathbf{p}, \mathbf{P})$ where d is positive (resp. negative) when $\mathbf{p}$ is on the right (resp. left) side of the directed path $\mathcal{P}$. Thus one have $d(0) > 0$ since the initial position of the robot is on the right side of the directed path. Then the absolute value of the signed distance is considered as the tracking error, i.e., $e(k) \triangleq |d(k)|$. The ray $l_{\mathbf{P}}$ starts from the projection point $\mathbf{P}$ and is tangent to the path $\mathcal{P}$ at $\mathbf{P}$, and the direction of the ray $l_{\mathbf{P}}$ is chosen to be coincide with the desired direction of the path following. Taking the idea of NLGL, when the robot is not far away from

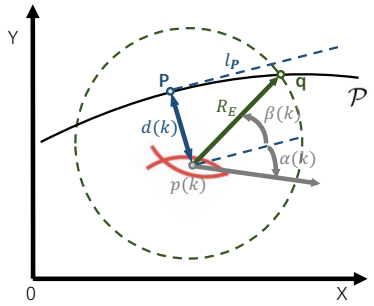


Fig. 3. The path-following task for a fish-like robot.

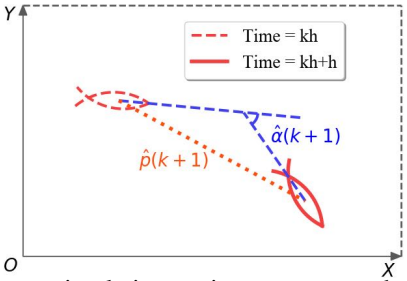


Fig. 4. Surrogate simulation environment maps the locomotion control input to the kinematics of the fish-like robot in the real system.

the path, a circle of radius R_E can be drawn at the robot's current position $\mathbf{p}(k)$, where $R_E > 0$ is constant. The circle, which can be seen as an exploration region of the robot, has two intersections points with the path $\mathcal{P}$, between which the one lying ahead of the robot is marked as $\mathbf{q}$. As shown in Fig.3, $\beta(k)$ is the angle between the ray $l_{\mathbf{P}}$ and the vector $\overrightarrow{\mathbf{pq}}$, and $\alpha(k)$ is the robot's current orientation relative to the ray $l_{\mathbf{P}}$. To quantitatively evaluate the control performance, we concern about three indicators, the average tracking error $\bar{e}$, the maximum tracking error e_{max}, and the overshooting times k_o, where $\bar{e}$ and e_{max} are the average and the maximum of the tracking error $e(k)$ during a task, respectively, k_o denotes the times when the tracking error exceeds the threshold R^*, and $R^*, 0 < R^* < R_B$, implies the satisfactory tracking range.

In this paper, considering the physical characteristics of the robotic fish that its body length is $0.443m$ and the average of its stable velocity is $0.5m/s$, we set the satisfactory tracking range $R^* = 0.1m$, and the exploration region $R_E = 0.643m$.

3.2 Surrogate Simulation Environment

To efficiently train a control policy in simulation and directly implement it on the real robot, a critical issue is to reduce the reality gap caused by the discrepancy between the simulation and the real system. However, the complexity of its hydrodynamic environment makes it impossible to construct a sufficiently accurate simulation environment with few computational resources using traditional methods like computational fluid dynamics.

To balance the accuracy and the computation speed, we built a surrogate simulation environment based on the experimental data of the real robot to introduce reality information, as well as to speed up the training process. Specifically, the surrogate model is mathematically formulated as a mapping function f_s

$$f_s : [t_a(k), u(k)] \rightarrow [\hat{p}(k+1), \hat{\alpha}(k+1), v(k+1)] \quad (4)$$

where $v(k) \in \mathbb{R}$ is the linear velocity of the robot at $t = kh$ and $v(0) = 0$, $\hat{p}(k+1) \in \mathbb{R}$ and $\hat{\alpha}(k+1) \in \mathbb{R}$ represent the variation of the position and that of the orientation of the

Table 1. Surrogate simulation environment

$t_a = 0/t_a = 1$	action m	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$
robot kinematics	$D^m_{t_a}$	0.14/0.13	0.15/0.16	0.15/0.15	0.15/0.17	0.15/0.16	0.14/0.16	0.16/0.16
	$A^m_{t_a}$	-0.26/-0.41	-0.10/-0.24	-0.06/-0.15	0.01/-0.04	0.11/0.11	0.20/0.13	0.24/0.38
	$v(k)$	$v(k) = (1 - \frac{1}{1.15^k})V_{max},\ V_{max} = 0.3$						
control input	X^m	-0.26/-0.26	-0.18/-0.18	-0.09/-0.09	0/0	0.18/0.18	0.26/0.26	0.35/0.35

fish during $t \in [kh, kh + h)$, respectively (Fig.4). The position of the fish is defined by its center of gravity $\mathbf{p}(k) \in \mathbb{R}^2$. We assumed $\hat{\alpha}(k) > 0$ in the counterclockwise direction. $t_a(k)$ and $u(k)$ denote the time tag and the control input as mentioned above, respectively. Considering the nonholonomic dynamics of the fish-like robot, we assumed that $v(k) \geq 0$, $\hat{p}(k) \geq 0$.

The concrete form of the mapping f_s in Eq. (4) was obtained based on the experimental data of the real fish-like robot. We selected M constants $X^m, m = 1, 2, \ldots, M$, where $\{X^1, X^2, \ldots, X^M\}$ are nearly uniformly distributed in the range $[U_{min}, U_{max}]$, so that X^m can be seen as the sampled data of the continuous domain $[U_{min}, U_{max}]$. Then, for each $u(k) = X^m$, we conducted the experiments G_r times for $k \in \{0, 1, 2, \ldots, K_r\}$. The average values of $\hat{p}(k + 1)$ when $t_a(k) = 0$ and when $t_a(k) = 1$ were calculated and recorded as $\tilde{D}^m_0$ and $\tilde{D}^m_1$, respectively. Similarly, the average values of $\hat{\alpha}(k + 1)$ when $t_a(k) = 0$ and when $t_a(k) = 1$ were calculated and recorded as A^m_0 and A^m_1, respectively. Next, we gained the linear velocity $v(k)$. We randomly chose the $u(k)$ in $[U_{min}, U_{max}]$ for each k, and performed the experiments G_C times for $k \in \{0, 1, 2, \ldots, K_C\}$. For each run, the value of the linear velocity is a curve from 0 up to its maximum value when t increasing. Thus we calculated the average of the G_C maximum values and recoded it as V_{max}, while the average of the G_C curves was figured and approximated as

$$v(k) = (1 - \frac{1}{1.15^k})V_{max}, \tag{5}$$

from which we obtained the $v(k)$ in the surrogate model (4). According to the linear velocity, we made some modification to the variation of the position obtained via the experiments that

$$D^m_{t_a} = \frac{v(k)}{V_{max}} \tilde{D}^m_{t_a},\ t_a = 0, 1. \tag{6}$$

To sum up, we represented the mapping f_s as

$$[D^m_{t_a}, A^m_{t_a}, v(k + 1)] = f_s[t_a(k), X^m]. \tag{7}$$

In this paper, considering the physical limitations of the robotic fish and the specific task which will be described later, we chose $[U_{min}, U_{max}] = [-0.5, 0.5]rad$, $M = 7$, $K_r = 23$, $G_r = 5$, $K_C = 59$, $G_C = 3$ and summarized the created surrogate model (7) in Table 1.

3.3 Formulating the control problem

We mathematically formulate the control problem for the robot as a discrete-time optimization problem. At each time step k, the fish-like robot observes the state s_k of its environment in a state space $\mathcal{S}$, and executes an action $a_k \in \mathcal{A}$ according to a policy $\pi(a_k|s_k)$, which is a mapping from state s_k to action a_k. Then the robot receives a scalar reward $r_k \in \mathcal{R}$, which measures how well it accomplishes the task, and the state changes to s_{k+1} according to the environmental dynamics of the surrogate simulation environment or that of the physical system. Given a trajectory $\tau(\pi)$ under policy π as $s_0, a_0, s_1, a_1, \ldots$ with the associated rewards $r_0, r_1, \ldots$, the infinite horizon discounted return along this trajectory is $\sum_{k=0}^{\infty} \gamma^k r_k$ where $\gamma \in (0, 1]$ is the discount factor. Then the control objective is to find an optimal policy π^* that maximizes the expectation of the discounted sum of rewards over an infinite horizon

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau(\pi)} \left[\sum_{k=0}^{\infty} \gamma^k r_k \right]. \tag{8}$$

In this study, the states s_k include the time tag $t_a(k)$, the robot's velocity $v(k)$, and some of its pose information selected according to the path-following task; the actions are the locomotion control input $u(k)$ to the fish-like robot; and the rewards are specified so as to complete the given task.

3.4 Deep Reinforcement Learning

To solve the above discrete-time optimization problem, reinforcement learning (RL) provides a possible way, in which the robot learns to act in a trial-and-error manner as to maximize the rewards obtained by interaction with its environment (Sutton and Barto (2018)). More specifically, the aim of the robot is to find an optimal policy which maximizes the reward so that the given task is well completed. For the control of autonomous fish-like robots in our study, we chose the A2C approach, which takes advantage of both the value-based RL methods and the policy-based RL methods, where the actor chooses the action based on the probability, and the critic evaluates the current policy to improve it. Specifically, the critic provides the actor with the advantage value $A(s)$ based on the value function $V(s)$, $A(s) = r + \lambda V(s') - V(s)$, and the actor updates the policy parameter using the gradient $\nabla_\theta \log \pi_\theta(s, a)A(s)$.

In this work, the A2C network is realized in a simple form where each of the actor- and critic-network has one fully connected hidden layer with 20 neurons. We trained the control policy within the surrogate simulation environment. To avoid over-fitting and to improve the generalization capabilities of the algorithm, the training process terminates when the average reward of the 50 episodes overreaches a task-specific threshold.

3.5 Observation, Action and Reward

Considering the path-following task, we defined the whole observations as $s_k = [t_a(k), v(k), d(k), \alpha(k), \beta(k)]$ at each time step k, where $t_a(k)$ and $v(k)$ are the time tag and the velocity of the robot, $d(k)$ is the distance from the robot location to the path, $\alpha(k)$ and $\beta(k)$ are the robot's orientation and the bearing angle relative to the path's projection (Fig.3). We chose $u(k)$, the locomotion control input of the robot, to be the action a_k which is the output of the control policy in our training method. The selected 7 actions are shown in Table 1. For the path-following task (Fig.3), we defined the reward as

$$r_k = \begin{cases} 1 - \frac{|d(k)|}{R_B}, & \text{when } |d(k)| < R_B \\ -1, & \text{when } |d(k)| \geq R_B \end{cases} \tag{9}$$

Obviously, r_k goes to 1 when the robot gets close to the desired path, i.e., $d(k) \to 0$, and r_k decreases when the distance from the robot location to the path $d(k)$ increases. If the fish swims out of the boundary described by the constant R_B, the reward r_k turns into -1 as a punishment.

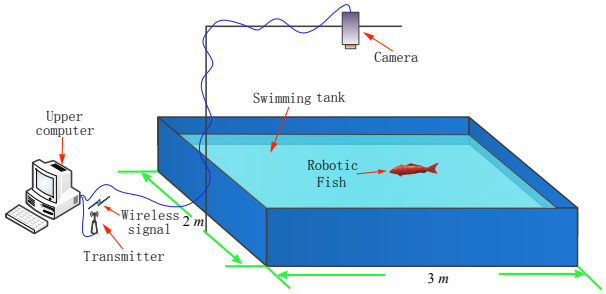


Fig. 5. Experimental platform of the fish-like robot.

3.6 Policy Training Details

For the given path-following task, it aims to control the fish-like robot to follow a randomly generated Bézier curve $\mathcal{P}$. To this end, we trained the robot to follow a straight line in surrogate simulation environment. At the beginning of each training episode, the robot was randomly located near the given line and on the right side of the line that the distance from the robot location to the path satisfied $0 < d(0) < R_E$ where $R_E = 0.643m$ as mentioned above, while the orientation of the robot $\alpha(0)$ is also set randomly. The training episode continues until the task is failed that the robot swims out of the boundary, i.e., $e(k) = |d(k)| \geq R_E$, or the time step arrives at its maximum which is set as $k_{max} = 100$, and this episode's reward $\bar{r}_e$ is defined as the average of the rewards r_k. After each training episode, the average reward of 50 latest episodes are calculated and marked as $\bar{r}_{e(50)}$. Then the next episode starts, and the loop continues until $\bar{r}_{e(50)}$ goes beyond the task-specsific threshold, which implies the situation when the robot's performance is good enough and is set as 0.94 for the path-following task.

4. EXPERIMENTAL EVALUATION

In this section, the training results and the experiments with real robots following Bézier curves are reported to show the effectiveness of our proposed approach.

4.1 Experimental Platform

To test the trained policy, we use an experimental platform consisting of a server computer, an overhead camera, a wireless communication module and a fish-like robot (Fig.5). The overhead camera captures images of a $3 \times 2m$ tank per $40ms$ and then sends them to the server computer, where images are processed to obtain the pose information of the robot. The upper computer can exchange information with robot through the wireless communication module. Thus, at each time step k, the upper computer collects the pose information of the robot and infers the whole observations $s_k = [t_a(k), v(k), d(k), \alpha(k), \beta(k)]$, and then the upper computer runs the training method and outputs an action $a_k = u(k)$, which is send to the robot to be executed through the wireless communication. We refer to (Wang et al. (2017); Yu et al. (2016)) for readers who are interested in more technical details about the experimental platform.

4.2 Experimental Results

As mentioned above, we trained the fish to follow a straight line in simulation. The training method and its parameters were adopt as described in Section 3.6. The simulation results of the

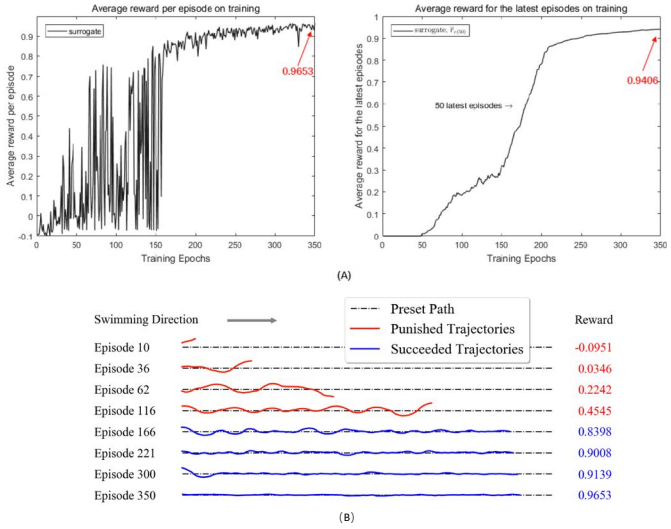


Fig. 6. Training to follow a straight line in surrogate simulation environment.

training process, as well as some typical episodes during training, are shown in Fig.6. Fig.6(A) shows the learning curves of the average reward of each episode and the average reward of 50 latest episodes in the surrogate environment. In the episode 350 within the surrogate environment, the fish fully followed the straight line and got the episode's reward of 0.9653, meanwhile, the average reward of 50 latest episodes $\bar{r}_{e(50)}$ increased as 0.9406 which reached the task-specific threshold 0.94 and thus a trained policy π_S^* was obtained. The training process for 350 episodes with the surrogate environment took only 50 minutes. To gain insight into the training process and the performance of our method, we selected some typical training episodes and revealed the fish's trajectories in Fig.6(B). The trajectory in red indicates that the fish swims out of the boundary ($|d(k)| \geq R_B$) that ends its corresponding episode, while the one in blue means the fish completes the task successfully during the episode.

Then, we directly deployed the trained policy π_S^* on the real fish-like robot to follow the Bézier curve. It's worth emphasizing that, in our training method, although the control policies are trained only in simulation and only using a straight line as the desired path, the trained policy is directly deployed on the real robot and performs well for a randomly generated Bézier curve. In this study, We chose $n = 6, 10, 8$ in Eq. (3), thus $7, 11, 9$ control points are involved and can be chose randomly to generate the desired curves $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ for the fish-like robot to follow with, respectively. Fig. 7 shows the control performance of our trained policy π_S^* on the real fish-like robot. For these three curves, we repeated the experiments five times. To quantitatively evaluate the performance of the trained policies, we mainly concerned about three indicators, $\bar{e}$, e_{max}, k_o (see Section 3.1), which were calculated and summarized in Table 2. Each data is the average of five runs.

Table 2. Deploying on the real fish-robot to follow the curves $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, respectively.

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$
$\bar{e}/e_{max}/k_o$	0.062/0.199/2.4	0.038/0.173/10.8	0.051/0.212/22

According to these experimental results, the maximum error is $0.212m$, the minimum error is $0.038m$. It's worth noting that the robotic fish is $0.443m$ long with the stable velocity of $0.5m/s$. That is to say, our method successfully tracked three

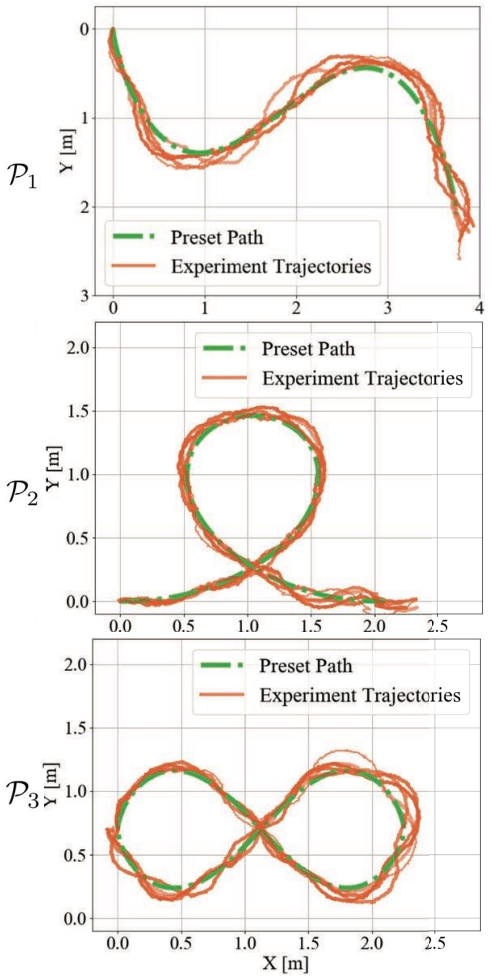


Fig. 7. Deploying on the real fish-like robot to follow randomly generated Bézier curve $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, respectively.

curves. At the same time, when $R^* = 0.1$, the overshooting times of $\mathcal{P}_1$ is 2.4, which shows that the trajectory has a certain degree of smoothness. Even for the complex curve $\mathcal{P}_3$, the overshooting times is only 22, which also shows the method is robust. Therefore, one could conclude that our trained policy is of superior performance.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a DRL based approach for path-following control of a fish-like robot in the real world. A modified CPG model has been designed for the locomotion control of the fish-like robot, so that the fish can achieve varied swimming behaviors by a single control input. Based on the experimental data of the real fish-like robot, we have built a surrogate simulation environment, within which the training accuracy and speed can be well balanced. For the path-following task, we have selected the A2C approach and trained the control policy in the surrogate simulation environment with a straight line as the desired path. Then the trained control policy has been directly implemented in a physical fish-like robot to validate its path-following capability. The experimental results have shown that the trained policies still performs well even with a randomly generated Bézier curve as the desired path. That is, our proposed approach has good practical applicability in controlling the fish-like robot in real world. To improve our results in this paper, we are working on designing a more accurate simulation environment to achieve more precise control.

REFERENCES

Butail, S., Abaid, N., Macrì, S., and Porfiri, M. (2015). Fish–robot interactions: robot fish in animal behavioral studies. In *Robot fish*, 359–377. Springer.

Fossen, T.I. and Pettersen, K.Y. (2014). On uniform semiglobal exponential stability (USGES) of proportional line-of-sight guidance laws. *Automatica*, 50(11), 2912–2917.

Ijspeert, A.J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4), 642–653.

Ijspeert, A.J., Crespi, A., Ryczko, D., and Cabelguen, J. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315, 1416–1420.

Kapitanyuk, Y.A., Proskurnikov, A.V., and Cao, M. (2018). A guiding vector-field algorithm for path-following control of nonholonomic mobile robots. *IEEE Transactions on Control Systems Technology*, 26(4), 1372–1385.

Lekkas, A.M. and Fossen, T.I. (2012). A time-varying lookahead distance guidance law for path following. *IFAC Proceedings Volumes*, 45(27), 398–403.

Li, L., Wang, C., and Xie, G. (2015). A general CPG network and its implementation on the microcontroller. *Neurocomputing*, 167, 299–305.

Oh, H., Kim, S., Shin, H.s., and Tsourdos, A. (2015). Coordinated standoff tracking of moving target groups using multiple UAVs. *IEEE Transactions on Aerospace and Electronic Systems*, 51(2), 1501–1514.

Ryuh, Y.S., Yang, G.H., Liu, J., and Hu, H. (2015). A school of robotic fish for mariculture monitoring in the sea coast. *Journal of Bionic Engineering*, 12(1), 37–46.

Shin, J., Kwak, D.J., and Lee, Y.i. (2017). Adaptive path-following control for an unmanned surface vessel using an identified dynamic model. *IEEE/ASME transactions on mechatronics*, 22(3), 1143–1153.

Streitlien, K., Triantafyllou, G.S., and Triantafyllou, M.S. (1996). Efficient foil propulsion through vortex control. *Aiaa journal*, 34(11), 2315–2319.

Sujit, P., Saripalli, S., and Sousa, J.B. (2014). Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, 34(1), 42–59.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Wang, C., Xie, G., Wang, L., and Cao, M. (2011). CPG-based locomotion control of a robotic fish: Using linear oscillators and reducing control parameters via PSO. *International Journal of Innovative Computing, Information and Control*, 7, 4237–4249.

Wang, C., Chen, X., Xie, G., and Cao, M. (2017). Emergence of leadership in a robotic fish group under diverging individual personality traits. *Royal Society open science*, 4(5), 161015.

Woo, J., Yu, C., and Kim, N. (2019). Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean Engineering*, 183, 155–166.

Yu, J., Wang, C., and Xie, G. (2016). Coordination of multiple robotic fish with applications to underwater robot competition. *IEEE Transactions on Industrial Electronics*, 63(2), 1280–1288.

Zhu, J., Wang, J., Zheng, T., and Wu, G. (2016). Straight path following of unmanned surface vehicle under flow disturbance. In *OCEANS 2016-Shanghai*, 1–7. IEEE.