Robotica (2024), **42**, pp. 947–961
doi:10.1017/S0263574723001613

RESEARCH ARTICLE

Reinforcement learning-based motion control for snake robots in complex environments

Dong Zhang, Renjie Ju and Zhengcai Cao

College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China
Corresponding author: Zhengcai Cao; Email: caozc@buct.edu.cn

Received: 18 July 2023; **Revised:** 23 October 2023; **Accepted:** 7 November 2023; **First published online:** 12 February 2024

Keywords: snake robot; path planning; motion planning

Abstract
Snake robots can move flexibly due to their special bodies and gaits. However, it is difficult to plan their motion in multi-obstacle environments due to their complex models. To solve this problem, this work investigates a reinforcement learning-based motion planning method. To plan feasible paths, together with a modified deep Q-learning algorithm, a Floyd-moving average algorithm is proposed to ensure smoothness and adaptability of paths for snake robots' passing. An improved path integral algorithm is used to work out gait parameters to control snake robots to move along the planned paths. To speed up the training of parameters, a strategy combining serial training, parallel training, and experience replaying modules is designed. Moreover, we have designed a motion planning framework consists of path planning, path smoothing, and motion planning. Various simulations are conducted to validate the effectiveness of the proposed algorithms.

1. Introduction
Inspired by biological snakes, engineered snake robots have slim bodies and multiple degree of freedoms (DOFs). They can move flexibly in narrow and complex environments [1]. Due to these features, they have good potential to replace human in some special work such as disaster rescuing and monitoring [2]. They have received great attention since the first one was designed by Hirose in 1972 [3]. Compared with wheeled robots, they are more difficult to control due to their complex mechanisms and special gaits [4].

In the past years, most researchers focused attention on the design, modeling, and gait control of snake robots. Path planning and path following are also necessary for snake robots to move in complex environment automatically [5]. Mathematical models are considered as bases of their path planning and following. In ref. [6], a unique mathematical model in which contact forces are mapped on the basis of a viscous friction model is presented. It predicts behaviors of a robot in a cluttered environment accurately. In ref. [7, 8], obstacle-aided locomotion strategies are researched for modular snake robots in cluttered environments. In ref. [9], Li designs an anti-sideslip line-of-sight method-based adaptive path following controller for snake robots. Later, he proposes a novel stable path-following method to fulfill formation control for multi-snake robots [10]. An obstacle-aided locomotion method is proposed by using piecewise helixes to improve the mobility of snake robots in messy environments in ref. [11]. In ref. [12], a trajectory planning method is proposed for dual robotic fingers to manipulate cartons in a complex folding process. These studies have advanced the state of the art of multi-DOF robots despite that their models are complex.

To simplify these models, a framework that can generate paths in low-dimensional work space and select generated gaits in a snake robot's shape space is proposed in ref. [13]. Typically, a trajectory optimization method is developed by using Lp-norm-based representations to encode collision constraints of robots and obstacle bounding boxes [14]. The method fails to deal with complex or unstructured environments. To stabilize constraint manifolds, an analytic smooth feedback control law is proposed in ref. [15]. A simplified snake robot model is presented for motion control [16]. Snake robots can move in multi-obstacle environments by using above methods based on accurate dynamic models. However, it is difficult to derive such models due to their complex mechanisms. Motion planning of snake robots includes two steps: path planning for their centers of mass (CMs) in workspace and the corresponding path following. Many path following methods of snake robots were researched in the past years [17–21]. In ref. [17], a guidance-based path following law is designed for them. A dynamic model and a trajectory tracking control law are proposed for them to track paths without singular configuration [18]. In ref. [19], an adaptive path following controller of snake robots is designed based on an improved Serpenoid curve. In ref. [20], an adaptive path following law is proposed and verified for snake robots. However, since snake robots move forward by swinging their bodies, it is difficult to control them moving along desired paths [21], as their trajectories are oscillation curves. It is required to calculate control parameters based on path following laws timely. These methods are time-consuming and inefficient.

Different from those model-based methods, suitable control parameters can be obtained without any prior experiences by using neural networks or reinforcement learning (RL) algorithms. In ref. [22], an energy-efficient snake gait policy is found by using deep reinforcement learning (DRL) algorithm. A deep deterministic policy gradient algorithm is used to tune a gait model and obtain optimal parameters of central pattern generator [23]. A two multi-layered spiking neural network is designed to achieve 3D gaits for snake robots to track certain moving objects in ref. [24]. Methods in ref. [22–24] are more effective than traditional empirical tuning process. However, they are not suitable for complex and changing environments. To improve environmental adaptability of snake robots, a DRL-based framework with a double deep Q-learning-based technique is proposed to learn the optimal policy for reaching goal points in unknown environments in ref. [25]. However, main variables of different unknown environments are frictions and stiffness but not obstacles. In ref. [26], RL is used to derive efficient and novel gaits for a terrestrial and aquatic 3-link snake robot. The strategy is verified, but the robot mechanism is simpler than typical multi-link snake robots.

In ref. [27], path following of snake robots is investigated by using approximate dynamic programming and neural networks. A path integral (PI) algorithm is suitable for motion planning due to its effective learning and stable numerical solution [28]. It is used in some learning tasks of different robots [29–31]. An extension of a PI policy improvement algorithm is proposed to adjust critical parameters automatically [29]. The algorithm is tested on three different types of simulated legged robots. In ref. [30], a policy improvement with PI is used to generate goal-directed locomotion of a snake robot with screw-drive units. In ref. [31], PI is used to compute gait parameters of a snake robot. However, some factors, for example, the arrival time and distance are not considered in these studies.

To plan passable paths and follow them without depending on accurate models, this work designs a motion planning framework for snake robots in multi-obstacle environments based on an RL algorithm. In order to plan a short collision-free path, we use a multi-objective fusion adaptive reward mechanism based on a DRL algorithm to design its loss function. Besides, weights of the neural network are updated by an experience replay technology with a gradient back-propagation method. To control a snake robot to follow the planned paths smoothly, we propose a path smooth algorithm by combining a Floyd algorithm [32] and a moving average (MA) algorithm. Gait parameters of snake robots can be computed by a PI algorithm. A parallel training strategy is designed to explore the control parameter space of global paths lightly.

The remainder of this article is organized as follows. A model and symbols of snake robots are given in Section 2. A DRL-based path planning algorithm and a path smoothing strategy are proposed in Section 3. A goal-oriented motion planning method based on PI is proposed in Section 4. Simulations are conducted in Section 5 to demonstrate successful path planning and moving through multi-obstacle environments. Conclusions are drawn in Section 6.

Table I. *Mathematical symbols.*

Symbol	Description
l	Half of the length of each link
m	Mass of each link
n	Number of links
g	The gravitational acceleration constant
$\boldsymbol{P}$	Position of the CM of the snake robot, $\boldsymbol{P} = [p_x, p_y]^T$
$\boldsymbol{p_i}$	Position of the CM of link i, $\boldsymbol{p_i} = [x_i, y_i]^T$
$\boldsymbol{X}, \boldsymbol{Y}$	$\boldsymbol{X} = [x_1, x_2, \dots, x_n]^T, \boldsymbol{Y} = [y_1, y_2, \dots, y_n]^T$
θ_i	Angle between link i and the XOZ plane
ϕ_i	Yaw angle of joint i, $\phi_i = \theta_{i+1} - \theta_i$
$\boldsymbol{C}_\theta, \boldsymbol{S}_\theta$	$\boldsymbol{C}_\theta = \text{diag}(\cos \boldsymbol{\theta}), \boldsymbol{S}_\theta = \text{diag}(\sin \boldsymbol{\theta})$

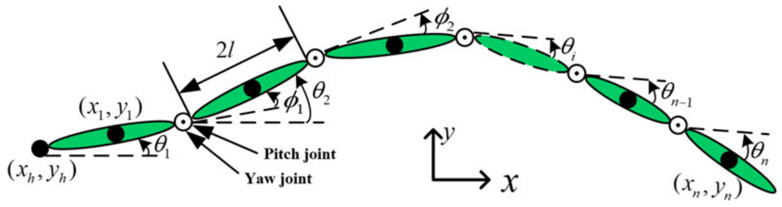

Figure 1. *Kinematic parameters of a n-link snake robot.*

2. Mathematical model of snake robots

To accurately control snake robots, we derive a kinematic model of a 7-link snake robot. Its parameters and mathematical symbols are shown in Fig. 1 and explained in Table I. The global frame position of the robot's CM is calculated as:

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{nm} \sum_{i=1}^{n} mx_i \\ \frac{1}{nm} \sum_{i=1}^{n} my_i \end{bmatrix} = \frac{1}{n} E \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{E}$ is a transition matrix defined as:

$$E = \begin{bmatrix} e^T & \boldsymbol{0}_{1\times n} \\ \boldsymbol{0}_{1\times n} & e^T \end{bmatrix}, \tag{2}$$

and $\boldsymbol{e} = [1, 1, \dots, 1]^T \in R^n$. Links must comply the following constraints:

$$x_{i+1} - x_i = lcos\theta_i + lcos\theta_{i+1}, \tag{3}$$

$$y_{i+1} - y_i = lsin\theta_i + lsin\theta_{i+1}. \tag{4}$$

Constraints of all links are as follows:

$$DX + lAcos\boldsymbol{\theta} = \boldsymbol{0}, \tag{5}$$

$$DY + lAsin\boldsymbol{\theta} = \boldsymbol{0}, \tag{6}$$

where

$$A = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix}, \tag{7}$$

$$D = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix}. \tag{8}$$

Velocities of all links can be computed as:

$$\dot{\boldsymbol{X}} = lK^T \boldsymbol{S}_\theta \dot{\boldsymbol{\theta}} + \boldsymbol{e}\dot{p}_x, \tag{9}$$

$$\dot{\boldsymbol{Y}} = -lK^T \boldsymbol{C}_\theta \dot{\boldsymbol{\theta}} + \boldsymbol{e}\dot{p}_y, \tag{10}$$

where $K = A^T(DD^T)^{-1}D$ and $T = \left[D^T, \frac{1}{n}\boldsymbol{e} \right]^T \in R^{n\times n}$.

3. Path planning of snake robots

Path planning is important for snake robots to move in multi-obstacle environments. Compared with traditional path planning methods, deep Q network (DQN) is more adaptive for different environments. An end-to-end mapping between passable path and environments can be established. Therefore, we use a DQN-based path planning method to improve snake robots' adaptability in complex environments.

3.1. Path planning based on DQN

It is well known that RL can be used for agents to maximize their cumulative rewards in unknown environments by exploring and exploiting collected experiences. In RL, exploration is contradictory to utilization. In the exploration, an agent experimenting with novel strategies may improve returns after long exploring. In the utilization, an agent focuses on maximizing rewards through actions. It is difficult to balance them, especially when an environment contains a huge state space or sparse rewards. An excessively large exploration rate may lead to endless exploration activities in some unexplored areas without any rewards. It thus sacrifices too much short-term benefits. An extremely large utilization rate may converge to suboptimal behaviors while ignoring action spaces with huge benefits.

Exploitation and utilization can be balanced explicitly by utilizing an ε-greedy policy with a random selected exploration rate $\varepsilon \in (0, 1)$. The snake robot chooses an action corresponding to the maximum Q value. For actions with same Q values, the robot randomly selects an action. With a probability of $1-\varepsilon$, the robot randomly selects an action in the action set.

The final strategy and convergence rate of the agent are determined by a reward function. In this work, the reward function is defined as:

$$r = \begin{cases} r_a, & collision \\ r_b, & collisionless \\ r_c, & others \end{cases}, \tag{11}$$

where r_a is a negative constant, while r_b and r_c are positive constants.

Algorithm 1: The DQN algorithm

```
Initialize a replay memory $D$ to the capacity $N$;
Initialize the action-value function $Q$ with random weights $\theta$;
Initialize the target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$;
for $E = 1, 2, \ldots, M$ do
    Initialize the sequence $s_1 = \{x_1\}$ and preprocess the sequence $\phi_1 = \phi(s_1)$;
    for $t = 1, 2, \ldots, T$ do
        Select a random action $a_t$ with a probability $\varepsilon$;
        Otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$;
        Execute an action $a_t$ in an emulator and observe a reward $r_t$ and an image $x_{t+1}$;
        Set $s_{t+1} = s_t, a_t, r_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$;
        Store the transition $(\phi_t, a_t, x_{t+1})$ in $D$;
        Sample random minibatch of transitions from $D$ $(\phi_j, a_j, r_j, \phi_{j+1})$;
            $y_j = \begin{cases} r_j, \text{ episode terminates at step } j+1 \\ r_j + \nu \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}), \text{ otherwise} \end{cases}$
        Perform gradient descent on $(y_j - Q(\phi_j, a_j; \theta))$ with respect to $\theta$;
        Reset $\hat{Q} = Q$ every $C$ steps;
    end
end
```

The pseudocode of the proposed DQN algorithm is shown in Algorithm 1. The algorithm has two key modules. The first one is a biology-inspired experience replay mechanism used to randomize the data. Effected by the mechanism, correlations in the observation sequence are removed and changes in the data distribution are smoothed. Updates of rewards and status obtained from each interaction are saved to update Q_θ.

With an empirical playback mechanism, the error between Q_θ and $\tilde{Q}$ is used to update parameters $\boldsymbol{\omega}$ of the neural network by using a gradient back-propagation method. An approximate Q value and a specific strategy can be obtained when w converges.

For the second module, an iterative strategy is used to update action values toward target values periodically. Correlations between them are deduced by this strategy. A snake robot selects and executes action a_t from the action set based on history experiences of a restore buffer and a specific strategy in the current state s_t. After action a_t, the robot switches to a new state s_{t+1} and obtains a reward r_{t+1}.

The snake robot repeats a process of exploration-learning-decision-utilization and gains experiences by interacting with environments. The experience replay technology is used to update its knowledge, that is, as historical experience to select the next action. The flow diagram of the algorithm is shown in Fig. 2. Its parameters are explained in Table II.

3.2. Path smoothing

Due to the simplification of action space, there are many redundant inflection points in the generated paths. The planned path points are not smooth enough for snake robots. To solve this problem, we adopt a Floyd-MA algorithm to smooth the planned paths.

The Floyd algorithm is used to select proper path points and MA used to smooth and interpolate the path between two neighbor path points. Points on the quadratic curve are selected as filtered results. New values of center points are computed by averaging their neighbor points, that is,

$$y_s(i) = \frac{1}{2n_p + 1}\left(y\left(i + n_p\right) + y\left(i + n_p - 1\right) + \ldots + y\left(i - n_p\right)\right), \tag{12}$$

Table II. *Parameters of the DQN-based path planning algorithm.*

Symbol	Description	Value
α	Learning rate	0.01
ϵ	Exploration rate	0.9
n_A	State feature dimension	2
B	Batch size	32
C	Step size	300
N	Capacity of restore buffer	500
E	The number of iterations	100,000

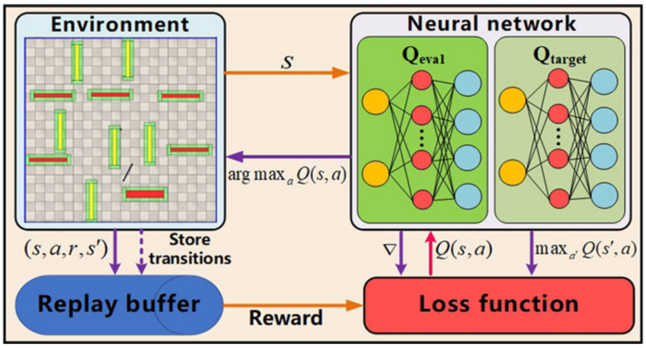

Figure 2. *The DQN-based path planning algorithm of snake robots.*

where n_p is the number of selected path points. Pseudocode of the Floyd method is shown in Algorithm 2. Its parameters are listed in Table III.

4. PI-based motion planning of snake robots

Motion planning of multiple DOFs snake robots can be defined as a stochastic optimal control problem. Inspired by ref. [31], we use a goal-oriented PI-based motion planning method to control a robot to move along the planned paths in complex environments. To plan goal-oriented motion according to the generated path points, the relationship between the stochastic optimal control and PI is analyzed based on a Bellman principle of optimality and a Hamilton–Jacobi–Bellman (HJB) equation. Different from ref. [31], the control difficulty and accuracy of the planned paths are considered in this work. Besides, a parallel training strategy is built to explore the control parameter space of the global path briskly.

4.1. Stochastic optimal control

Stochastic optimal control aims at maximized goal performance. To evaluate performances of the trajectory τ_i between t_i and t_N), a reward function is defined as:

$$J(\tau_i) = \phi[\boldsymbol{x}(t_N), t_N] + \int_{t_i}^{t_N} L[\boldsymbol{x}(t), \boldsymbol{u}(t), t]dt, \tag{13}$$

where $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are the state and control vector at time t, respectively. Parameters $\phi[\,\cdot\,]$ and $L[\,\cdot\,]$ are a terminal reward and an immediate reward at time t, respectively. The minimum cost function in stochastic optimal control is as follows:

$$V_{t_i} = \min_{u_{t_i}:t_N} E_{\tau_i}[J(\tau_i)], \tag{14}$$

Algorithm 2: The Floyd algorithm

```
Input: The sequence of point $P$ generated by a $Q$ network
Output: The sequence of screened point $P_n$
1  Remove adjacent collinear path points;
2  for $i = n_p - 3, \ldots, 0$ do
3      only if;
4      if $\boldsymbol{p_{i-1}p_i}$ and $\boldsymbol{p_ip_{i+1}}$ are collinear vectors then
5          Remove the connecting path point $p_i$;
6      end
7  end
8  Remove excess turning points;
9  for $i = n_p - 3, \ldots, 0$ do
10     for $j = 0, \ldots, i - 1$ do
11         if There is no collision between path points $p_i$ and $p_j$ with obstacles then
12             for $k = i - 1, \ldots, j$ do
13                 Remove path point $p_k$;
14             end
15         end
16     end
17 end
18 Return $P_n$
```

Table III. Parameters of the path smoothing algorithm.

Weights	y_{-3}	y_{-2}	y_{-1}	y_0	y_1	y_2	y_3
$\hat{y}_{-3}$	32/42	15/42	3/42	−4/42	−6/42	−3/42	5/42
$\hat{y}_{-2}$	15/42	12/42	9/42	6/42	3/42	0/42	−3/42
$\hat{y}_{-1}$	3/42	9/42	12/42	12/42	9/42	3/42	−6/42
$\hat{y}_0$	−4/42	6/42	12/42	14/42	12/42	6/42	−4/42
$\hat{y}_1$	−6/42	3/42	9/42	12/42	12/42	9/42	3/42
$\hat{y}_2$	−3/42	0/42	3/42	6/42	9/42	12/42	15/42
$\hat{y}_3$	5/42	−3/42	−6/42	−4/42	3/42	15/42	32/42

where $E_{\tau_i}[\,\cdot\,]$ is expectations of all trajectories starting at state $\boldsymbol{x}_{t_i}$. The stochastic dynamic system is considered as:

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_t, t) + \boldsymbol{G}(\boldsymbol{x}_t)(\boldsymbol{u}_t + \boldsymbol{\varepsilon}_t) = \boldsymbol{f}_t + \boldsymbol{G}_t(\boldsymbol{u}_t + \boldsymbol{\varepsilon}_t), \tag{15}$$

where $\boldsymbol{x}_t \in R^{n\times 1}$ is a state vector of the system, $\boldsymbol{G}_t = \boldsymbol{G}(\boldsymbol{x}_t) \in R^{n\times p}$ is a control matrix, $\boldsymbol{f}_t = \boldsymbol{f}(x_t, t) \in R^{n\times 1}$ is a passive dynamics, $\boldsymbol{u}_t \in R^{p\times 1}$ is a control vector, and $\boldsymbol{\varepsilon}_t \in R^{p\times 1}$ is a Gaussian noise with a zero mean and a variance Σ_ε. The immediate reward is defined as:

$$L_t = L[\boldsymbol{x}(t), \boldsymbol{u}(t), t] = q_t + \frac{1}{2}\boldsymbol{u}_t^T \boldsymbol{R} \boldsymbol{u}_t, \tag{16}$$

where $q_t = q(\boldsymbol{x}_t, t)$ is an arbitrary state-dependent cost function, and $\boldsymbol{R}$ is a positive semi-definite weight matrix of the quadratic control cost. The HJB equation of this stochastic optimal control problem is defined as:

$$-\partial_t V_t = \min\left(L_t + (\nabla_x V_t)^T \boldsymbol{F}_t\right) + \frac{1}{2} trace\left((\nabla_{xx} V_t)\, \boldsymbol{G}_t \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} \boldsymbol{G}_t^T\right), \tag{17}$$

where ∇_x and ∇_{xx} are the Jacobian matrix and Hessian matrix, respectively. Substituting the optimal control into (17), we can get a second-order nonlinear partial differential equation (PDE):

$$-\partial_t V_t = q_t + (\nabla_x V_t)^T f_t - \frac{1}{2}(\nabla_x V_t)^T \boldsymbol{G}_t^T \boldsymbol{R}^{-1} \boldsymbol{G}_t^T (\nabla_x V_t) + \frac{1}{2} trace\left((\nabla_{xx} V_t)\, \boldsymbol{G}_t \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} \boldsymbol{G}_t^T\right). \tag{18}$$

4.2. Transformation of HJB into a linear PDE

Due to the complexity of HJB equation, an exponential transformation of the value function is proposed as $V_t = -\lambda \lg \Psi_t$. A second-order linear PDE can be obtained as:

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \boldsymbol{f}_t^T (\nabla_x \Psi_t) + \frac{1}{2} trace\left((\nabla_{xx} \Psi_t)\, \boldsymbol{G}_t \Sigma_\varepsilon \boldsymbol{G}_t^T\right), \tag{19}$$

where $\Psi_{tN} = \exp\left(-\frac{1}{\lambda}\phi_{tN}\right)$ is the boundary condition, and the analytic solution can be obtained according to the Feyman–Kac theorem:

$$\Psi_{t_i} = E\left(\Psi_{t_i} \exp\left(-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt\right)\right) = E\left(\Psi_{t_i}\left(\exp\left(-\frac{1}{\lambda}\phi_{tN} - \frac{1}{\lambda}\int_{t_i}^{t_N} q_t dt\right)\right)\right). \tag{20}$$

In this way, the stochastic optimal control problem is transformed into an approximate PI problem:

$$\Psi_{t_i} = \lim_{dt\to 0} \int p(\boldsymbol{\tau}_i | \boldsymbol{x}_i) \exp\left[-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt\right)\right] d\boldsymbol{\tau}_i, \tag{21}$$

where $\boldsymbol{\tau}_i = [\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i+1}}, \ldots, \boldsymbol{x}_{t_N}]^T$ is a sample path which starts from $\boldsymbol{x}_{t_i}$. The probability of $\boldsymbol{\tau}_i$ is as follows:

$$p(\boldsymbol{\tau}_i | \boldsymbol{x}_i) = \frac{\exp\left(-\frac{1}{\lambda}\right)}{\int \exp\left(-\frac{1}{\lambda}\right) d\boldsymbol{\tau}_i}. \tag{22}$$

The optimal controls can be simplified as:

$$\boldsymbol{u}_{t_i} = \int p(\boldsymbol{\tau}_i | \boldsymbol{x}_i) \boldsymbol{u}_L(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i. \tag{23}$$

However, the above analytic solutions depend on the system model $\boldsymbol{f}_t$. Using the PI algorithm, the probability of trajectory $\tau_{i,k}$, that is, the k-th path generated randomly in the start state $\boldsymbol{x}_{t_i}$, is updated as:

$$P(\tau_{i,k}) = \frac{\exp\left(-\frac{1}{\lambda} S(\tau_{i,k})\right)}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} S\left(\tau_{i,j}\right)\right)}. \tag{24}$$

The control vector can be iterated as:

$$\boldsymbol{u} = \boldsymbol{u} + \sum_{k=1}^{K} P(\tau_{i,k}) \frac{\boldsymbol{R}^{-1} \boldsymbol{G}_{t_i,k} \boldsymbol{G}_{t_i,k}^T}{\boldsymbol{G}_{t_i,k}^T \boldsymbol{R}^{-1} \boldsymbol{G}_{t_i,k}}. \tag{25}$$

The objective is to find $\boldsymbol{u}_i$ that minimizes the cost function corresponding to the trajectory τ_i generated by the control variable:

$$V = \min_u E_{\tau_i}[R(\tau_i)] = \min_u \int R(\tau_i) P(\tau_i) d\tau_i. \tag{26}$$

The probability of trajectory τ_i is computed as:

$$P(\tau_i) = \frac{S(\tau_i)}{\sum_{j=1}^{K} \tau_i}, \tag{27}$$

where K is the number of generated trajectories and $S(\tau_i)$ is computed as:

$$S(\tau_i) = \exp\left(-\frac{R(\tau_i) - \min(\boldsymbol{R})}{\max(\boldsymbol{R}) - \min(\boldsymbol{R})}\right), \tag{28}$$

where $\boldsymbol{R} = [R(\tau_1), R(\tau_2), \cdots, R(\tau_K)]$. The control vector $\boldsymbol{u}^*$ of the next iteration is as follows:

$$\boldsymbol{u}^* = \sum_{i=1}^{K} P(\tau_i)\boldsymbol{u}_i. \tag{29}$$

4.3. Policy improvements with PI

A compound serpenoid curve templet is proposed based on research of biological snakes in ref. [33]. For the lateral undulation gait, snake robots move forward by swinging their yaw joints. Joint angles are defined as:

$$\phi_{i,ref} = \alpha_h \sin(\omega_h t + (i-1)\beta_h) + \gamma_h, \tag{30}$$

where $\phi_{i,ref}$ is the reference yaw angle of joint i at time t, and α_h, ω_h, β_h, and γ_h are amplitude, angular frequency, phase difference, and phase offset, respectively. Gait parameters of the robot can be expressed as a vector $\boldsymbol{U} = [\alpha_h, \omega_h, \beta_h, \gamma_h]^T$. Values of initial and the ξ-th iteration of gait parameters are denoted as $\boldsymbol{U_0} = [\alpha_0, \omega_0, \beta_0, \gamma_0]^T$ and $\boldsymbol{U_\xi} = [\alpha_\xi, \omega_\xi, \beta_\xi, \gamma_\xi]^T$, respectively. Each iteration of training generates K paths randomly. Path τ_i is determined by $\boldsymbol{u_i} = [\varepsilon_{\alpha_{h_i}}, \varepsilon_{\omega_{h_i}}, \varepsilon_{\beta_{h_i}}, \varepsilon_{\gamma_{h_i}}]^T$, where ε obeys normal distribution with a zero mean. The corresponding loss function $R(\tau_i)$ is expressed as:

$$R(\tau_1) = \sqrt{(x_0 - x_g)^2 + (y_0 - y_g)^2}. \tag{31}$$

During the goal-oriented movement of multiple path points, the snake robot may fail to track unsmooth paths. In order to make the robot complete the goal-oriented motion, we design a new loss function that combines the deviation of trajectory and moving difficulty:

$$R(\tau_1) = \sqrt{(x_0 - x_g)^2 + (y_0 - y_g)^2} + \left|\arctan\bar{\theta} - \frac{y_{g+1} - y_g}{x_{g+1} - x_g}\right|, \tag{32}$$

where $\bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i$ is a robot's forward direction. Parameters of path i in the $(\xi + 1)$-*th* training are as follows:

$$\boldsymbol{U^i_{\xi+1}} = \boldsymbol{U_\xi} + \boldsymbol{u_i} = \left[\alpha_\xi + \varepsilon_{\alpha_i}, \omega_\xi + \varepsilon_{\omega_i}, \beta_\xi + \varepsilon_{\beta_i}, \gamma_\xi + \varepsilon_{\gamma_i}\right]^T. \tag{33}$$

Gait parameters after $\xi + 1$ iterations are updated as:

$$\boldsymbol{U_{\xi+1}} = \boldsymbol{U_\xi} + \boldsymbol{u^*}. \tag{34}$$

The variance of ε is related to the loss function value produced by $\boldsymbol{U_\xi}$ after ξ iterations, and the standard deviation in the $(\xi + 1)$-*th* training is

$$\boldsymbol{\sigma} = \left[\sigma_\alpha, \sigma_\omega, \sigma_\beta, \sigma_\gamma\right]^T = \left[f_\alpha(r), f_\omega(r), f_\beta(r), f_\gamma(r)\right]^T. \tag{35}$$

4.4. Closed-loop control framework

To realize the autonomous movement of snake robots in complex environments, this work designs a closed-loop control framework. As shown in Fig. 3, it is composed of path planning, path smoothing, and motion planning modules.

Algorithm 3 The Goal-oriented Motion Planning Algorithm Based on PI Method

```
Initialize the controls of gait equation $\boldsymbol{U}_0$;
Initialize the variance of stochastic control increment $\boldsymbol{\sigma}^2$;
for $\xi = 1, 2, \cdots, M$ do
    for $i = 1, 2, \cdots, K$ do
        Generate the stochastic control increment $\boldsymbol{u}_i$ according to $\sigma^2$;
            $$\boldsymbol{U}_{\xi+1}^i = \boldsymbol{U}_\xi + \boldsymbol{u}_i$$
        Generate trajectory $\tau_i$ and loss $R(\tau_i)$ according to $\boldsymbol{U}_{\xi+1}^i$;
    end
    Update the controls $\boldsymbol{U}_{\xi+1}$;
    Generate trajectory $\tau_i$ and corresponding loss function $r$ according to $\boldsymbol{U}_{\xi+1}$;
    Update the variance $\boldsymbol{\sigma} = \boldsymbol{f}(r)$;
end
```

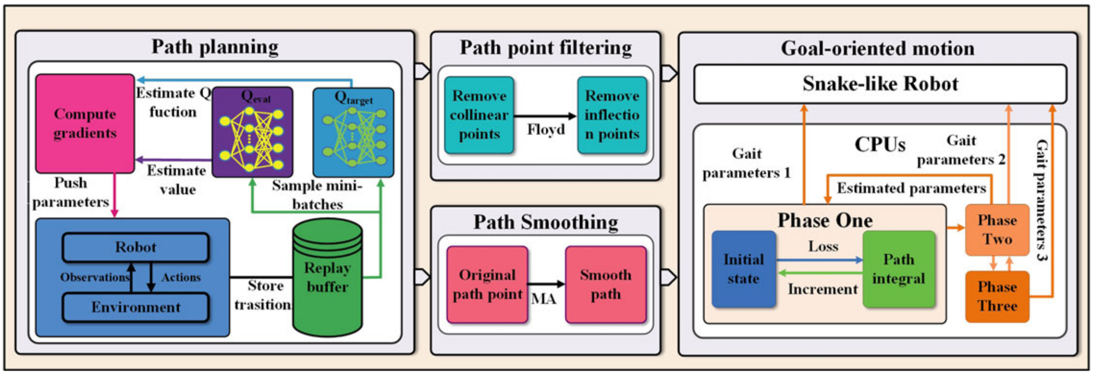

Figure 3. *Global framework for motion planning of snake robots.*

In path planning, actions are selected by a neural network. The robot gains experiences by continuous interacting with environments. Weights of the network are updated by a back-propagation algorithm based on an experience replay technology. In path smoothing, path points generated by the trained strategy are filtered and adjusted by the proposed Floyd-MA path smoothing algorithm.

In motion planning, obtained reliable path points are used to obtain gait parameters trained by PI algorithm in parallel to achieve goal-oriented motion. Its pseudocode is shown in Algorithm 3.

5. Simulation

To test the proposed DQN-based path planning and PI-based motion planning algorithms, this work conducts extensive simulations in CoppeliaSim and Pycharm.

5.1. Path planning of snake robots

A virtual snake robot prototype is built in CoppeliaSim. Same with our previous work [20], a robot prototype is constructed. To generate anisotropic friction between a robot and the ground, a pair of passive wheels are equipped at each link. Communication and control of snake robots in CoppeliaSim are implemented through a remote API.

Obstacles are represented by rectangles with arbitrary poses. They are dilated on each side. The dilated distance is the maximum gait amplitude set in the optimization problem. Parameters of the neural

Table IV. *Parameters of the PI motion planning algorithm.*

Symbol	Description	Value
M	Number of iterations	20
K	Number of paths generated in each iteration	30
S_t	Number of training	600
(x_0, y_0)	Start position	(-2.3, -0.5)
(x_g, y_g)	Target position	(3.1, -3.1)
$\boldsymbol{U}_0$	Initial values of gait parameters	$[0.6, 7, 0.7, 0]^T$
$\boldsymbol{\sigma}_0$	Standard deviation of normal distribution	$[0.025, 0.5, 0.1, 0.0]^T$

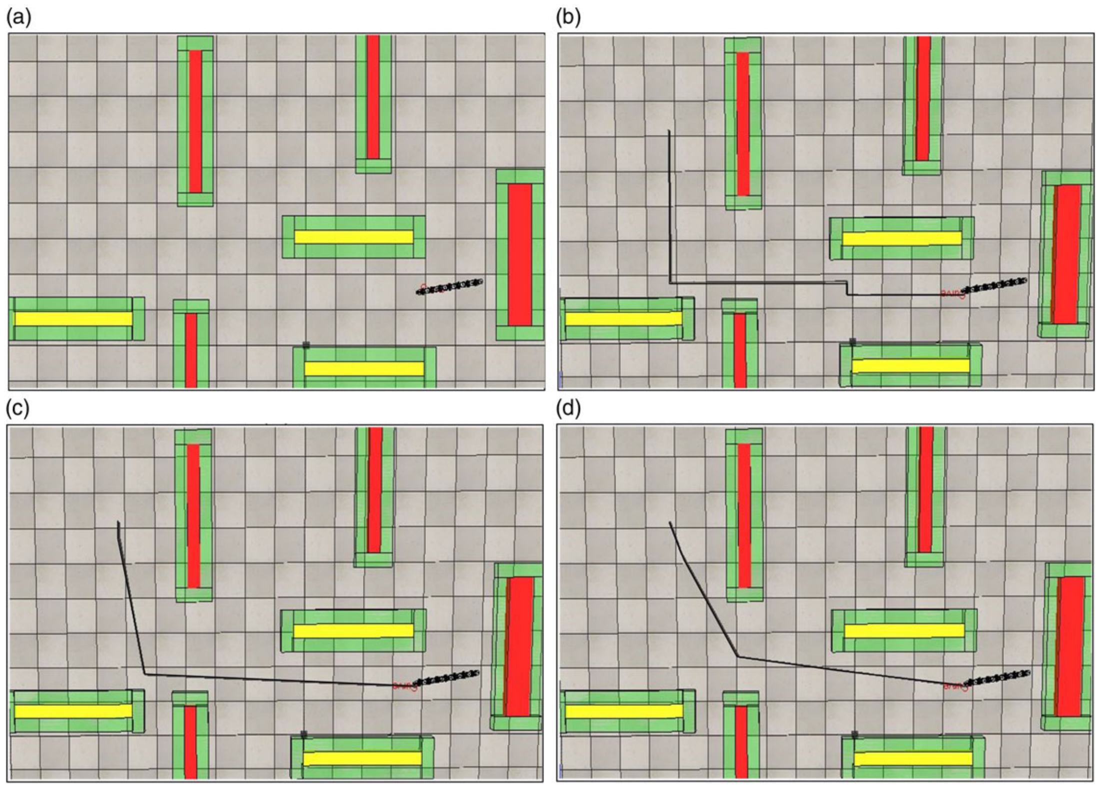

Figure 4. *Path planning of a snake robot. (a) Simulation environment; (b) path planning; (c) path point filtering; and (d) path smoothing.*

network and a path smoothing algorithm are set as Table I and Table II, respectively. Simulation results are exhibited to show the process of motion planning.

As shown in Fig. 4(a), obstacles (red and yellow) are dilated by safety margins (green). The start position of a snake robot is the lower right corner. The target position is the upper left corner. As shown in Fig. 4(b), a passable path in a complex environment is generated by using DQN, but there exist collinear path points and excessive curvature changes between neighboring points. As shown in Fig. 4(c), redundant collinear path points and points with excessive curvature changes are deleted by the Floyd algorithm. As shown in Fig. 4(d), MA improves the path smoothness and reduces the difficulty of path following.

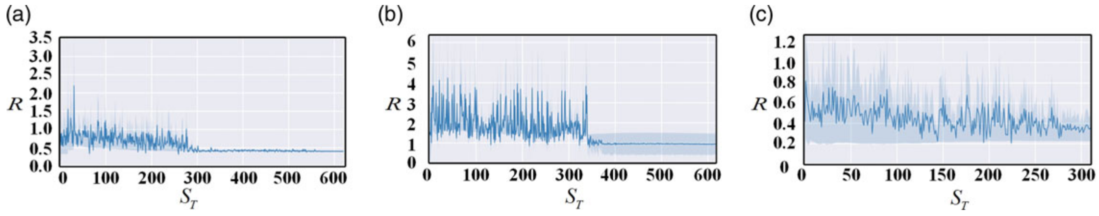

Figure 5. *Performances of PI in goal-oriented motion of the snake robot. (a) The first stage. (b) The second stage. (c) The third stage.*

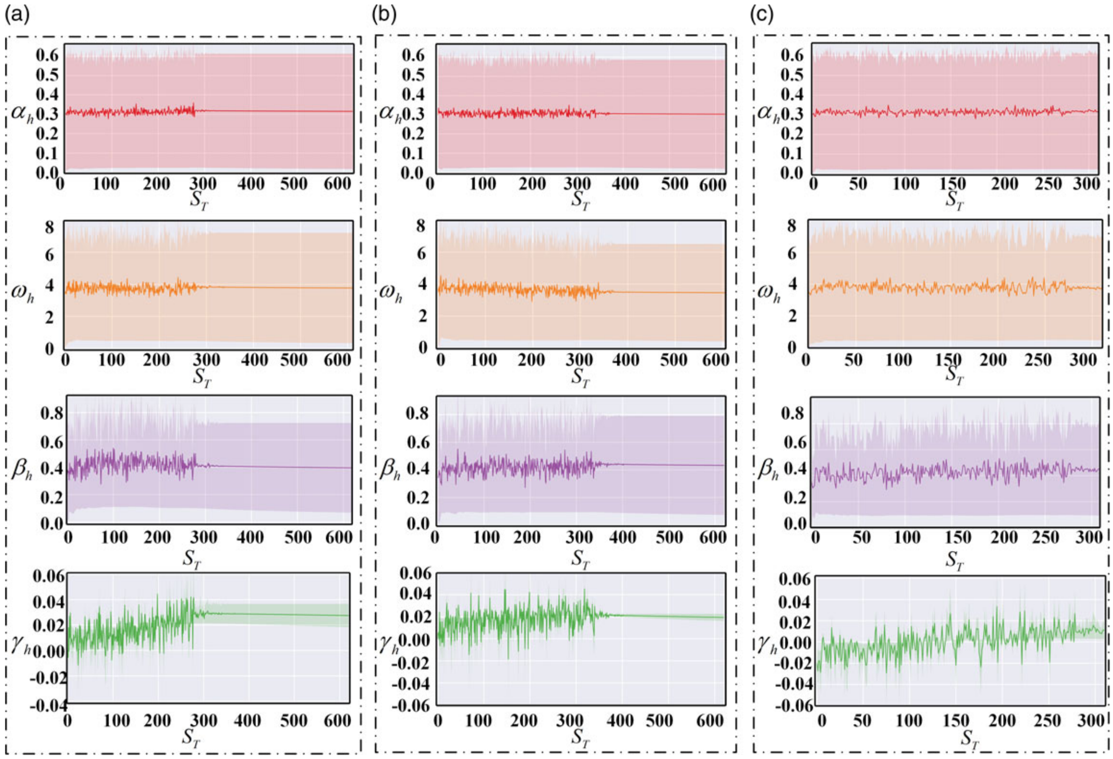

Figure 6. *Values of gait control parameters. (a) The first stage. (b) The second stage. (c) The third stage.*

5.2. Motion planning of snake robots

In order to verify the feasibility of the proposed PI-based motion planning algorithm, we conduct extensive simulation in CoppeliaSim. The PI method is used to train three path segments in parallel as prior knowledge, and then the spaces of gait parameters on the global path are explored serially to realize end-to-end mapping between the planned path points and gait parameters. Parameters used in motion planning are listed in Table IV. The standard deviation is computed as:

$$\sigma = f(r) = \begin{cases} \sigma_0, & r \geq 0.5 \\ r\sigma_0/2, & 0.2 \leq r < 0.5 \\ r\sigma_0/4, & 0.05 \leq r < 0.2 \\ r\sigma_0/8, & 0.2 \leq r < 0.05 \end{cases} . \tag{36}$$

As shown in Fig. 5(a)-(c), loss functions of the first stage to the third stage converge to stable states. As shown in Fig. 6(a)-(c), corresponding to the change curve of the parameters of the gait equation

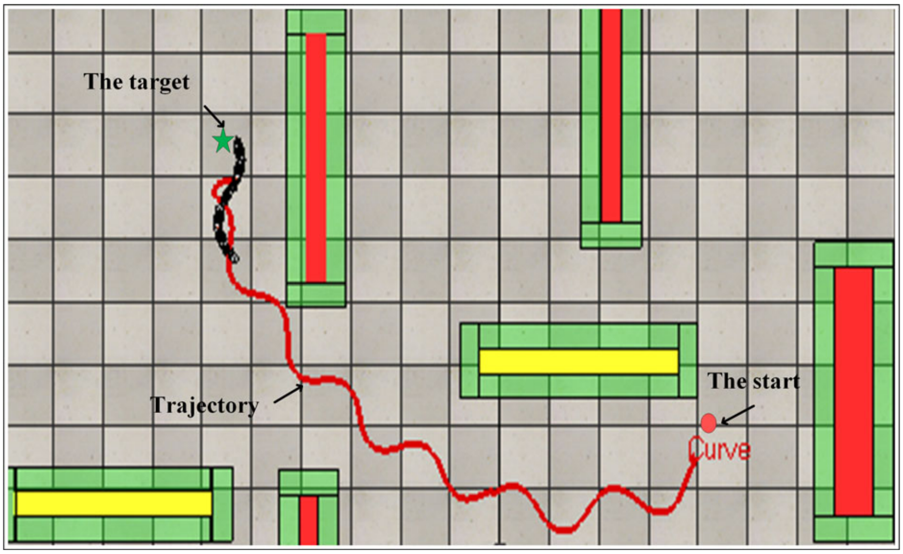

Figure 7. *A snake robot moves in a multi-obstacle environment controlled by the PI-based motion planning algorithm.*

as the iteration count increases. These curves converge to stable states from initial states quickly. After training, gait parameters in the first stage converge to $\boldsymbol{U}^{(1)} = [0.6129, 7.2016, 0.7238, 0.0367]^T$ as shown in Fig. 6(a). Gait parameters in the second stage converge to $\boldsymbol{U}^{(2)} = [0.5869, 6.5121, 0.7915, 0.0227]^T$ as shown in Fig. 6(b). Gait parameters in the third stage converge to $\boldsymbol{U}^{(3)} = [0.5914, 6.5964, 0.8163, 0.0381]^T$ as shown in Fig. 6(c).

Trajectory of the robot's CM is shown in Fig. 7. It can be got that the snake robot can move toward to a target smoothly in a multi-obstacle environment controlled by using the proposed path planning and motion planning methods.

6. Conclusion

This work proposes an RL-based motion planning framework for snake robots. A DQN method is used to plan a passable path in multi-obstacle environment. A multi-objective fusion adaptive reward mechanism is designed for the method. For the mechanism, weights of DQN are updated based on an experience replay technology. A Floyd-MA path smoothing algorithm is proposed to smooth the planned path. To control snake robots to move along the planned paths, a PI algorithm is used to compute gait parameters. Several local paths are in parallel trained to gather experiences based on a PI method to minimize the global deviation from the designed path. Experimental results show that the proposed motion planning algorithm can control snake robots to move autonomously in multi-obstacle environments.

Our future work intends to investigate their motion control methods by combining the path planning, path following, and environmental perception [34, 35] in complex environments with dynamic obstacles.

Author contributions. Dong Zhang, Renjie Ju, and Zhengcai Cao conceived and designed the study and wrote the article.

Financial support. This work is supported by the National Natural Science Foundation of China (52105005), the China Postdoctoral Science Foundation (2021M690320), and State Key Laboratory of Robotics and System (SKLRS-2022-KF-15).

Competing interests. The authors declare no competing interests exist.

Ethical approval. The authors declare none.

References

[1] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl and J.T. Gravdahl, “A review on modelling, implementation, and control of snake robots,” *Robot Auton Syst* **60**(1), 29–40 (2012).

[2] J. K. Hopkins and S. K. Gupta, “Design and modeling of a new drive system and exaggerated rectilinear-gait for a snake-inspired robot,” *J Mech Robot* **6**(2), 021001 (2014).

[3] S. Hirose. *Biologically Inspired Robots: Snake-Like Locomotors and Minipulators* (Oxford University Press, England,1993).

[4] M. Tanaka and K. Tanaka, “Control of a snake robot for ascending and descending steps,” *IEEE Trans Robot* **31**(2), 511–520 (2015).

[5] G. Wang, W. Yang, Y. Shen and H. Shao, “Adaptive Path Following of Snake Robot on Ground With Unknown and Varied Friction Coefficients,” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain (IEEE, 2018), 7583–7588.

[6] A. Singh, C. Gong and H. Choset, “Modelling and Path Planning of Snake Robot in Cluttered Environment,” In: *2018 International Conference on Reconfigurable Mechanisms and Robots*, Delft, Netherlands (IEEE, 2018) pp. 1–6.

[7] P. Liljebäck, K. Y. Pettersen, Ø Stavdahl and J. T. Gravdahl, “Hybrid modelling and control of obstacle-aided snake robot locomotion,” *IEEE Trans Robot* **26**(5), 781–799 (2010).

[8] P. Liljebäck, K. Y. Pettersen, O. Stavdahl and J. T. Gravdahl. *Snake Robots: Modelling, Mechatronics, and Control: Advances in Industrial Control* (Springer Verlag, London, 2012). pp. 8–17.

[9] D. Li, B. Zhang, P. Li, E. Q. Wu, R. Law, X. Xu, A. Song and L. Zhu, “Parameter estimation and anti-sideslip line-of-sight method-based adaptive path-following controller for a multi-joint snake robot,” *IEEE Trans Syst Man Cyber Syst* **53**(8), 4776–4788 (2023).

[10] D. Li, B. Zhang, R. Law, E. Q. Wu and X. Xu, “Error-constrained formation path-following method with disturbance elimination for multi-snake robots,” *IEEE Trans Ind Electron* **71**(5), 4987–4998(2024). doi: 10.1109/TIE.2023.3288202.

[11] T. Takanashi, M. Nakajima, T. Takemori and M. Tanaka, “Obstacle-aided locomotion of a snake robot using piecewise helixes,” *IEEE Robot Auto Lett* **7**(4), 10542–10549 (2022).

[12] H. Liu and J. Dai, “An approach to carton-folding trajectory planning using dual robotic fingers,” *Robot Auton Syst* **42**(1), 47–63 (2003).

[13] R. Hatton, R. Knepper, H. Choset, D. Rollinson, C. Gong and E. Galceran, “Snakes on a Plan: Toward Combining Planning and Control,” In: *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany (IEEE, 2013) pp. 5174–5181,

[14] H. C. Alexander, P. H. Nak-seung, I. V. Erik and A. V. Patricio, “Optimal Trajectory Planning and Feedback Control of Lateral Undulation in Snake-Like Robots,” In: *Annual American Control Conference (ACC)*, Milwaukee, WI, USA (IEEE, 2018) pp. 2114–2120 .

[15] E. Rezapour, K. Y. Pettersen, P. Liljebäck and J. T. Gravdahl, “Path Following Control of Planar Snake Robots Using Virtual Holonomic Constraints,” In: *IEEE International Conference on Robotics and Biomimetics*, Shenzhen, China (IEEE, 2013) pp. 530–537.

[16] H. Fukushima, T. Yanagiya, Y. Ota, M. Katsumoto and F. Matsuno, “Model predictive path-following control of snake robots using an averaged model,” *IEEE Trans Contr Syst Tech* **29**(6), 2444–2456 (2021).

[17] E. Rezapour and P. Liljebäck, “Path following control of a planar snake robot with an exponentially stabilizing joint control law,” *IFAC Proceed Vol* **46**(10), 28–35 (2013).

[18] F. Matsuno and H. Sato, “Trajectory Tracking Control of Snake Robots Based on Dynamic Model,” In: *IEEE International Conference on Robotics and Automation*, Barcelona, Spain (IEEE, 2005) pp. 3029–3034.

[19] D. Li, Z. Pan, H. Deng and L. Hu, “Adaptive path following controller of a multijoint snake robot based on the improved serpenoid curve,” *IEEE Trans Ind Electron* **69**(4), 3831–3842 (2022).

[20] Z. Cao, D. Zhang and M. Zhou, “Direction control and adaptive path following of 3-D snake-like robot motion,” *IEEE Trans Cyber* **52**(10), 10980–10987 (2022).

[21] P. Liljebäck, K. Y. Pettersen, O. Stavdahl and J. T. Gravdahl, “Controllability and stability analysis of planar snake robot locomotion,” *IEEE Trans Automat Contr* **56**(6), 1365–1380 (2011).

[22] Y. Liu and A. Farimani, “An energy-saving snake locomotion gait policy using deep reinforcement learning (2021). doi: 10.48550/arXiv.2103.04511 *arXiv*.

[23] K. Qiu, H. Zhang, Y. Lv, Y. Wang, C. Zhou and R. Xiong, “Reinforcement Learning of Serpentine Locomotion for a Snake Robot,” In: *IEEE International Conference on Real-time Computing and Robotics*, Xining, China (IEEE, 2021) pp. 468–473.

[24] Z. Jiang, R. Otto, Z. Bing, K. Huang and A. Knoll, “TargetTracking Control of a Wheel-Less Snake Robot Based on a Supervised Multi-Layered SNN,” In: *IEEE/RAS International Conference on Humanoid Robots*, Las Vegas, NV, USA (IEEE, 2020) pp. 7124–7130.

[25] S. Khan, T. Mahmood, S. Ullah, K. Ali and A. Ullah, “Motion Planning for a Snake Robot Using Double Deep Q-Learning,” In: *International Conference on Artificial Intelligence*, Islamabad, Pakistan (IEEE, 2021) pp. 264–230.

[26] J. Shi, T. Dear and S. Kelly, “Deep reinforcement learning for snake robot locomotion,” *IFAC-PapersOnLine* **53**(2), 9688–9695 (2020).

[27] Z. Cao, Q. Xiao, R. Huang and M. Zhou, “Robust neuro-optimal control of underactuated snake robots with experience replay,” *IEEE Trans Neur Net Lear Syst* **29**(1), 208–217 (2018).

[28] E. Theodorou, J. Buchli and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *J Mach Learn Res* **11**(104), 3137–3181 (2010).

[29] K. Yamamoto, R. Ariizumi, T. Hayakawa and F. Matsuno, “Path integral policy improvement with population adaptation,” *IEEE Trans Cyber* **52**(1), 312–322 (2022).

[30] S. Chatterjee, T. Nachstedt, F. Worgotter, M. Tamosiunaite, P. Manoonpong, Y. Enomoto, R. Ariizumi and F. Matsuno, "Reinforcement Learning Approach to Generate Goal-Directed Locomotion of a Snake-like Robot with Screw-Drive Units," In: *International Conference on Robotics in Alpe-Adria-Danube Region*, Smolenice, Slovakia (IEEE, 2014) pp. 1–7.

[31] Y. Fang, W. Zhu and X. Guo, "Target directed locomotion of a snake like robot based on path integral reinforcement learning," *Patt Recog Art Intell* **32**(1), 1–9 (2019).

[32] Y. Zhao, G. Wu, F. Gui, C. Xu and Z. Xie, "Optimal coordination path selecting method for conduction transformation based on floyd algorithm," *Procedia Comput Sci* **162**, 227–234 (2019).

[33] J. Dai, M.TraversT. Dear, C. Gong and H. Choset, "Robot-Inspired Biology: The Compound-Wave Control Templat," In: *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA (IEEE, 2015) pp. 5879–5884,

[34] Z. Cao, J. Li, D. Zhang, M. Zhou and A. Abusorrah, "A multi-object tracking algorithm with center-based feature extraction and occlusion handling," *IEEE Ttans Intell Transp* **24**(4), 4464–4473 (2023).

[35] H. Xie, D. Zhang, J. Wang, M. Zhou, Z. Cao, X. Hu and A. Abusorrah, "Semi-direct multimap slam system for real-time sparse 3-d map reconstruction," *IEEE Trans Instru Measure* **72**(4502013), 1–13 (2023).

Cite this article: D. Zhang, R. Ju and Z. Cao (2024). "Reinforcement learning-based motion control for snake robots in complex environments", Robotica **42**, 947–961. https://doi.org/10.1017/S0263574723001613