Research Article

Path-Integral-Based Reinforcement Learning Algorithm for Goal-Directed Locomotion of Snake-Shaped Robot

Qi Yongqiang , Yang Hailan , Rong Dan , Ke Yi, Lu Dongchen , Li chunyang ,
and Liu Xiaoting

China University of Mining and Technology (CUMT), School of Mathematics, 221116 Xuzhou, China

Correspondence should be addressed to Qi Yongqiang; qiyongqiang3@163.com

Received 6 August 2020; Revised 3 January 2021; Accepted 20 January 2021; Published 8 February 2021

Academic Editor: A. E. Matouk

Copyright © 2021 Qi Yongqiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a goal-directed locomotion method for a snake-shaped robot in 3D complex environment based on path-integral reinforcement learning. This method uses a model-free online Q-learning algorithm to evaluate action strategies and optimize decision-making through repeated “exploration-learning-utilization” processes to complete snake-shaped robot goal-directed locomotion in 3D complex environment. The proper locomotion control parameters such as joint angles and screw-drive velocities can be learned by path-integral reinforcement learning, and the learned parameters were successfully transferred to the snake-shaped robot. Simulation results show that the planned path can avoid all obstacles and reach the destination smoothly and swiftly.

1. Introduction

Research in the field of snake-shaped robots has been carried out for several decades [1–3]. This is because snake-shaped robots can not only be used as experimental platforms [4, 5] but also be employed for searching and rescuing operations, such as mine disaster rescue, search, and detection of people buried after the earthquake [6]. The snake-shaped robots are multimodule serially connected robots with no fixed base by imitating the snake’s body structure and movement mode.

Generally speaking, the snake-shaped robot consists of several segments connected in a serial manner as shown in Figure 1, and the main form of movement is undulation movements, which imitate real snake movements [7, 8]. In this paper, the snake-shaped robot is composed of screw-drive units with passive wheels, and the screw-drive units are serially connected through active joints. With this structure, the snake-shaped robot can move in any direction by appropriately adjusting the angular velocity of screw-drive units.

The problem of robot path planning in an unknown environment has attracted more and more researchers’ attention. The main research methods include the genetic algorithm, the gradient algorithm, the nonlinear mapping

algorithm, the linear mapping algorithm, and so on [9, 10]. These traditional approaches are improved to solve the path planning problem. Liljeback obtains the deviation of the centroid of the snake-shaped robot from the expected path through real-time monitoring. This deviation is used as a joint control parameter to adjust the head movement direction so that the snake-shaped robot follows the desired straight path [11–16]. Rezapour uses the desired angle of the head joint to adjust the swing of the head link, so that the centroid of the snake-shaped robot follows the desired straight path [17]. Hasanabadi uses sensors to measure the distance between the robot and the desired path to calculate the desired angle, and the angle is used as the joint control parameter to realize that the robot follows a straight path and a circle without side slip path [18]. Matsuno introduces a functional model of the desired path to establish the controller, so that the robot head follows the desired circular path [19]. Mohammadi controls the velocity components in all directions to make the snake robot’s center of mass follow a circular path, but it requires that the path function is known [20]. Ariizumi established a cost function based on the expected path and head trajectory and used the cost function as the control parameter to realize the snake-shaped robot following a sinusoidal path [21].

In order to make the snake-shaped robot reach the destination safely and smoothly, the first task is to find proper control parameters for generating goal-directed locomotion. This is a complex and challenging task, and we apply a special type of reinforcement learning, called path-integral reinforcement learning combining the time series analysis method for the task. Path-integral reinforcement learning is selected because this method can stably solve the “dimensional disaster” problem [22] and has proved successful for different robot learning tasks [23].

This paper proposes a collision-free goal-directed locomotion method based on path-integral reinforcement learning in a three-dimensional environment. The novelty of this paper and the main contributions of this work are as follows: a path-integral reinforcement learning model is established, which describes in detail how the robot can accumulate experience from historical data to optimize decision-making. The Q table is updated based on the prior information and provided to the robot as historical experience, which greatly improves the learning efficiency of the robot. To describe the influence of collision avoidance, the perturbation matrix is introduced and the planned path can avoid obstacle safely and reach the destination eventually. The simulation results show that the planned paths can avoid all obstacles smoothly and swiftly and reach the destination eventually.

The rest of the paper is organized as follows: Section 2 describes the kinematic model of the snake-shaped robot. Section 3 focuses on the path-integral-based reinforcement learning algorithm. Section 4 explains goal-directed locomotion of the snake-shaped robot. The simulation results are given in Section 5. Section 6 concludes the paper.

2. The Description of Kinematic Model of Snake-Shaped Robot

The kinematic model of the snake-shaped robot [7] can be described as

$$A\dot{\omega} = Bu, \quad (1)$$

where $\omega = [x, y, \psi, \phi_1, \phi_2, \phi_3, \dots, \phi_{n-1}]^T$ is the state vector and $u = [\theta_1, \theta_2, \theta_3, \dots, \theta_n, \phi_1, \phi_2, \phi_3, \dots, \phi_{n-1}]^T$ is the control input vector. (x, y) is the position of the head of the snake-shaped robot. ψ is the orientation of the first screw-drive unit with respect to (x, y) and is automatically adjusted with the snake-shaped robot kinematics. $\phi_1, \phi_2, \phi_3, \dots, \phi_{n-1}$ are the yaw joint angles which are relative orientations of one screw unit to the previous one. $\theta_1, \theta_2, \theta_3, \dots, \theta_n$ are the angular velocities in corresponding screw-drive units from the head, respectively. $\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dots, \dot{\phi}_{n-1}$ are the angular velocities of the joints starting from ϕ_1 which is the angular velocity of the first joint from the head. A and B are the system matrices and depend on system configurations such as screw-drive unit radius, inclination of blade, and length of one screw-drive unit [7].

The learning of locomotion control parameters such as screw velocities and joint angles can be carried out as follows. Suppose that the initial position of the head of the

snake-shaped robot is at (x_0, y_0) and the target position of the snake-shaped robot is (x_g, y_g) . The two parameter vectors used to represent the control policy can be described as follows:

$$U = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dots, \dot{\theta}_n, \phi_1, \phi_2, \phi_3, \dots, \phi_{n-1}]. \quad (2)$$

The snake-shaped robot's locomotion control policy following the kinematic model in equation (1) can be represented by U of equation (2). After learning final values of U , the snake-shaped robot will start from the initial position (x_0, y_0) and then move toward the desired goal (x_g, y_g) .

We know that the cost function or the objective function is the key component that affects the convergence of learning. In this article, we use Euclidean distance as a cost function as follows:

$$C(x, y) = \sqrt{(x - x_g)^2 + (y - y_g)^2}. \quad (3)$$

The cost function provides the distance between the reached head position (x, y) and the target position (x_g, y_g) of the snake-shaped robot. It can be seen that when the cost function is almost zero, the learned parameter vector will reach the final value. That is to say, the snake-shaped robot reaches the goal and completes the task. Therefore, with the final parameter vector, the robot will move toward the given target point.

In addition, the influence of external noise on path-integral reinforcement learning needs to be considered. As an open parameter in the learning process, external noise needs to be designed in a task-based manner. For path-integral reinforcement learning in this paper, a random value ε is selected by us from the normal distribution $N(0, 1)$, and ε can be dynamically adjusted according to the noise-free cost C_{t-1} obtained at the end of the previous update. Then, the external noise interference can be expressed as follows:

$$e_{m,k} = \begin{cases} \frac{\varepsilon}{9} e^{-(1/C_{t-1})}, & C_{t-1} > 5, \\ 0.05\varepsilon, & 1 \leq C_{t-1} \leq 5, \\ 0.025\varepsilon, & 0 < C_{t-1} < 1, \end{cases} \quad (4)$$

where m is the number of update, k is the number of noisy rollout, and $e_{m,k}$ is the external noise interference during k -th noisy trajectory or rollout of the m -th update.

Suppose that the noise applied to screw velocities is $e_{\dot{\theta}_1(m,k)}, e_{\dot{\theta}_2(m,k)}, \dots, e_{\dot{\theta}_n(m,k)}$, respectively. Similarly, assume that the noise applied to joint angles is $e_{\phi_1(m,k)}, e_{\phi_2(m,k)}, \dots, e_{\phi_{n-1}(m,k)}$, respectively. All these noise distributions follow above description of the external noise interference in every noisy trajectory $\tau_{m,k}$, but their patterns are different in a trajectory as ε is a random value.

The main steps for path-integral reinforcement learning applied to goal-directed locomotion of the snake-shaped robot are as follows: firstly, the parameter vector U can be selected according to the learning task. Secondly, the number

of rollouts K per update needs to be fixed and $K = 45$ in this paper. According to the parameter vector U to be learned, in

every rollout k , the trajectory with noisy parameters is simulated starting from robot start position (x_0, y_0) :

$$\tau_{m,k} \left(\dot{\theta}_1 + e_{\dot{\theta}_1(m,k)}, \dot{\theta}_2 + e_{\dot{\theta}_2(m,k)}, \dots, \dot{\theta}_n + e_{\dot{\theta}_n(m,k)}, \phi_1 + e_{\phi_1(m,k)}, \phi_2 + e_{\phi_2(m,k)}, \dots, \phi_{n-1} + e_{\phi_{n-1}(m,k)} \right), \quad (5)$$

where $\dot{\theta}_i + e_{\dot{\theta}_i(m,k)}$, ($i = 1, 2, \dots, n$) represent the noisy of screw velocity parameters and $\phi_j + e_{\phi_j(m,k)}$, ($j = 1, 2, \dots, n-1$) denote the noisy of joint angles. Trajectory of the snake-shaped robot can be generated by using the robot kinematic model described as equation (1) and the parameters of locomotion control such as screw velocities and joint angles. It can be seen that both the noisy screw velocities and the noisy joint angles as required for learning U are given in equation (5). According to the reached position $(x_{m,k}, y_{m,k})$ of the snake-shaped robot's head at the end of this rollout, then the final cost for this rollout can be calculated by evaluating the cost function as follows: $I_{m,k} = C(x_{m,k}, y_{m,k})$. With this method, all K noisy rollouts from start position of the snake-shaped robot within one update process m can be completed and corresponding $I_{m,k}$ can be stored. Then, the exponential value can be calculated for each rollout as follows:

$$S(\tau_{m,k}) = \exp^{-45 \left(I_{m,k} - \min_k \{I_{m,k}\} / \max_k \{I_{m,k}\} - \min_k \{I_{m,k}\} \right)}. \quad (6)$$

Then, the probability weighting P_m^k for each rollout can be calculated as follows:

$$P_m^k = \frac{S(\tau_{m,k})}{\sum_{i=1}^K S(\tau_{m,i})}. \quad (7)$$

Then, corrections for the parameter vector can be calculated based on P_m^k :

$$\begin{aligned} \Delta \dot{\theta}_i &= \sum_{k=1}^K P_m^k e_{\dot{\theta}_i(m,k)}, \\ \Delta \dot{\phi}_i &= \sum_{k=1}^K P_m^k e_{\dot{\phi}_i(m,k)}. \end{aligned} \quad (8)$$

The complete update vector can be constructed according to equation (8):

$$\Delta U = \left[\Delta \dot{\theta}_1, \Delta \dot{\theta}_2, \Delta \dot{\theta}_3, \dots, \Delta \dot{\theta}_n, \Delta \dot{\phi}_1, \Delta \dot{\phi}_2, \Delta \dot{\phi}_3, \dots, \Delta \dot{\phi}_{n-1} \right]. \quad (9)$$

Then, the control vector U_m at the end of an update m can be calculated as $U_m = U + \Delta U$.

3. Path-Integral-Based Reinforcement Learning Algorithm

The path-integration algorithm is used to solve the stochastic optimal control problem, and its purpose is to minimize the performance index of the stochastic dynamic system. The

analysis and derivation of stochastic optimal control is based on the optimal Bellman principle and the Hamilton-Jacobi-Bellman (HJB) equation. In this paper, we first analyze the stochastic optimal control of the mobile robot and then use the generalized path-integral control method to obtain the numerical solution of the stochastic dynamic system.

3.1. Stochastic Dynamical System. In general, the stochastic dynamic system can be expressed as follows [10]:

$$\dot{X}_t = f(X_t, t) + G(X_t)(u_t + \varepsilon_t) = f_t + G_t(u_t + \varepsilon_t), \quad (10)$$

where $X_t \in \mathbb{R}^{n \times 1}$ denotes the system state, $G_t \in \mathbb{R}^{n \times p}$ is the control matrix, $f_t \in \mathbb{R}^{n \times 1}$ represents the passive dynamics, $u_t \in \mathbb{R}^{p \times 1}$ stands for the control vector, and $\varepsilon_t \in \mathbb{R}^{p \times 1}$ is the Gaussian noise with variance $\Sigma_\varepsilon \in \mathbb{R}^{p \times p}$ and zero expectation. For mobile robots, the general stochastic dynamic system can be expressed by the following equation:

$$\begin{cases} \dot{x}_t = v \cos \varphi_t, \\ \dot{y}_t = v \sin \varphi_t, \\ \dot{\varphi}_t = \omega_t + \varepsilon_t, \end{cases} \quad (11)$$

where v and ω_t are the linear and angular velocities, (x_t, y_t) denotes the position of the robot on the 2-D plane, and φ_t represents its orientation, and $X_t = [x_t, y_t, \varphi_t]^T$, $f_t = [v \cos \varphi_t, v \sin \varphi_t, 0]^T$, $G_t = [0, 0, 1]^T$, $u_t = \omega_t$, and $\varepsilon_t = \varepsilon_t$. This paper studies the path planning of the snake robot based on the path-integration RL algorithm. Therefore, the linear velocity v remains constant, and only the angular velocity ω_t is used as a control to reduce the parameters in the RL, thereby accelerating the training convergence process.

3.2. Stochastic Optimal Control. For a trajectory τ_i with a start time of t_i and an end time of t_f , its performance function is defined as follows:

$$J(\tau_i) = \phi_{t_f} + \int_{t_i}^{t_f} L[X_t, u_t, t] dt, \quad (12)$$

where $\phi_{t_f} = \phi(X_{t_f}, t_f)$ is a terminal value and $\int_{t_i}^{t_f} L[X_t, u_t, t] dt$ is a process value. The immediate cost can be defined as

$$L_t = L[X_t, u_t, t] = q_t + \frac{1}{2} u_t^T R u_t, \quad (13)$$

where $q_t = q(X_t, t)$ is the cost function related to the state and R is the positive semidefinite weight matrix of the

quadratic control cost. The goal of stochastic optimal control is to find the control u_t to minimize the following performance criterion function:

$$V_t = \min_{u_{t_i:t_f}} E_{\tau_i} [J(\tau_i)], \quad (14)$$

where $E_{\tau_i}[\cdot]$ is the expectation of all trajectories starting from X_{t_i} . In order to solve the noiseless optimal control problem,

$$-\partial_t V_t = \min_{u_{t_i:t_f}} \left(q_t + \frac{1}{2} u_t^T R u_t + (\nabla_x V_t)^T F_t + \frac{1}{2} \text{trace}((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^T) \right), \quad (15)$$

where $\nabla_x V_t$ and $\nabla_{xx} V_t$ are the corresponding Jacobian and Hessian matrices and F_t is defined as $F_t = f(X_t, t) + G(X_t)u_t$. The corresponding optimal

there are two main methods: one is to use the Ponteria Gold Minimal Principle (PMP), which requires solving a pair of ordinary differential equations, and the other is to use partial differential equations (PDEs) of the HJB equation. For the stochastic dynamic system with noise shown in equation (10), the extension of the HJB method for stochastic optimal control is expressed as

control can be calculated as $u_t = -R^{-1}G_t^T(\nabla_{x_t} V_t)$ [10]. Then, a nonlinear second-order PDE can be obtained as follows:

$$-\partial_t V_t = q_t + (\nabla_x V_t)^T f_t - \frac{1}{2} (\nabla_x V_t)^T G_t R^{-1} G_t^T (\nabla_x V_t) + \frac{1}{2} \text{trace}((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^T). \quad (16)$$

V_t can be transformed into exponential form $V_t = -\lambda \log \Psi_t$, where λ is a positive constant, and then use $\lambda R^{-1} = \Sigma_\epsilon$ [24, 25]; then, PDE (16) can be simplified to the following form with the boundary condition which is $\Psi_{t_f} = e^{-q_{t_f}/\lambda}$:

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + f_t^T (\nabla_x V_t) + \frac{1}{2} \text{trace}((\nabla_{xx} \Psi_t) G_t \Sigma_\epsilon^T G_t^T). \quad (17)$$

3.3. Simplification of Path-Integral-Based RL. It is very difficult to obtain the analytical solution of PDE (17), so we replace it with commonly used numerical solutions, and one of which uses the Feynman–Kac theorem [26] as follows:

$$\Psi_{t_i} = E_{\tau_i} \left(\Psi_{t_f} e^{-\int_{t_i}^{t_f} (1/\lambda) q_t dt} \right) = E_{\tau_i} e^{\left(-(1/\lambda) \Psi_{t_f} - (1/\lambda) \int_{t_i}^{t_f} q_t dt \right)}. \quad (18)$$

After discretization, the stochastic optimal control problem can be approximately transformed into the following path-integration problem:

$$\Psi_{t_i} = \lim_{dt \rightarrow 0} \int p(\tau_i | x_i) \exp \left[-\frac{1}{\lambda} \left(\phi_{t_f} + \sum_{j=i}^{K-1} q_{t_j} dt \right) \right] d\tau_i, \quad (19)$$

where $\tau_i = (x_{t_i}, \dots, x_{t_f})$ is a sample trajectory with its probability as $p(\tau_i | x_i)$. After some analysis [27], the following numerical solutions of optimal control can be obtained:

$$u_{t_i} = \int p(\tau_i | x_i) u(\tau_i) d\tau_i, \quad (20)$$

where $u(\tau_i) = (u_{t_i}, \dots, u_{t_f})$ is used to generate the trajectory τ_i and the probability is defined as

$$p(\tau_i | x_i) = \frac{e^{-(1/\lambda)S(\tau_i)}}{\int e^{-(1/\lambda)S(\tau_i)} d\tau_i}, \quad (21)$$

where $S(\tau_i)$ is the general version of the path cost defined in [25]. However, the analytical form of calculating the path cost is very time-consuming because it involves many matrix inversions. Considering this fact, in order to simplify the calculation and speed up the training, we choose the linear path cost $S(\tau_i)$ function:

$$S(\tau_i) = J(\tau_i) = t_f - t_i. \quad (22)$$

It can be clearly seen that the generalized path-integral control method using the numerical solution of the stochastic dynamic system can avoid the calculation of gradient and kinematic inverse, ensuring fast and reliable convergence.

4. Goal-Directed Locomotion Algorithm

In the process of snake-shaped robot path-integral reinforcement learning, the reward that the snake-shaped robot gets depends on its own actions. Therefore, this paper extends the Markov decision process (MDP) model of path-integral reinforcement learning to multiple Markov decision processes (MDPs). Suppose that the snake-shaped robot can choose m actions ($a_i, i = 1, 2, \dots, m$), and the number of states of the snake-shaped robot is k ($s_j, j = 1, 2, \dots, m$). Based on the basic model of path-integral reinforcement learning, combined with task goal, this paper defines a snake-shaped robot goal-directed locomotion learning framework as shown in Figure 2.

It can be seen from Figure 2 that in order to improve the learning speed of the robot, first of all, environmental factors are preprocessed to eliminate some irrelevant environmental factors, and then a three-dimensional environmental model is established. Then, the prior information is updated to the knowledge base, which improves the learning efficiency. In this model, based on the current state S_t , the snake-shaped robot takes an action A_t in the action set according to a certain policy rule based on the historical experience of the knowledge base. Next, when the action is executed, the environment will be transformed into a new state S_{t+1} , and a new reward value R_{t+1} will be obtained. Finally, the historical experience is updated through continuous learning and further the knowledge base is improved. The snake-shaped

robot periodically interacts with the environment, repeating the process of “exploration-learning-decision-utilization,” learning from historical actions, and updating their knowledge base as historical experience to guide the next action selection.

4.1. Guidelines for Setting the Reward Function. The reward function defines the snake-shaped robot’s learning goal and determines its perception state based on the environment; that is, it defines the value of the action. Since the robot tries to maximize the reward, the reward function is essentially used to guide the robot to achieve its goal. Therefore, the setting of the reward function will determine the convergence speed and degree of the reinforcement learning algorithm. In this article, the reward function is defined in the form of sparse reward, as shown in the following:

$$R = \begin{cases} R_1, & \text{when the robot collides with an obstacle,} \\ R_2, & \text{when there is a collision between robots,} \\ R_3, & \text{when the robot reaches the target point,} \\ 0, & \text{other situations,} \end{cases} \quad (23)$$

where R_1, R_2, R_3 are constants and $R_1 < 0, R_2 < 0, R_3 > 0$.

4.2. Guidelines for Setting the Update Value Function. The goal-directed locomotion of the snake-shaped robot uses the Q-learning algorithm; that is to say, after determining all the environment states S and actions A , a matrix Q will be generated, and the element $Q(S, A)$ in the matrix Q represents the value of robot’s choice of action A_t under the environment state S_t .

The process of updating the value function is as follows: in the environment state S_t , when the snake-shaped robot selects an action A_t according to the action selection strategy, the snake-shaped robot reaches a new environment state S_{t+1} after executing the action. At this time, we update the value of $Q(S, A)$ in the matrix Q . The update formula of $Q(S, A)$ is as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)], \quad (24)$$

where γ is the attenuation coefficient and α is the learning efficiency.

4.3. Guidelines for Setting the Action Selection. In the path-integral reinforcement learning problem, exploration and utilization are a pair of contradictions: exploration means that the robot must try different behaviors and then collect more information, but utilization is that the robot makes the best decision under the current information. Exploration may sacrifice some short-term benefits and obtain more

long-term and accurate benefit estimates by collecting more information; utilization focuses on maximizing short-term benefits based on the information already available. The exploration cannot be carried out endlessly, otherwise, too much short-term benefits will be sacrificed, and the overall benefits will be damaged; at the same time, short-term benefits should not be valued too much while ignoring some unexplored behaviors that may bring huge benefits. In this paper, ϵ -greedy exploration is used, where ϵ is the probability of the snake-shaped robot randomly selected ($0 < \epsilon < 1$). In the case of probability $1 - \epsilon$, the snake-shaped robot selects

an action corresponding to the maximum Q value. If there is multiple Q with the same value, an action is randomly selected. When the probability is ε , the snake-shaped robot randomly selects an action from the action set.

4.4. Guidelines for Setting the Collision Avoidance. Collision avoidance must be considered in the process of the goal-directed locomotion, to simplify the calculation process, the ideal velocity for robot is assumed to be v_T , and then the actual velocity v_d can be calculated as follows:

$$v_d = P(u, v)v_T, \quad (25)$$

where $P(u, v)$ is the improved perturbation matrix [28], and the actual velocity v_d is calculated by modifying the target velocity v_T . Then, the planned path can be obtained by the recursive integration of v_d . To describe the influence of collision avoidance, the perturbation matrix $P(u, v)$ is defined as follows:

$$P(u, v) = I - \frac{n_q n_q^T}{D^*(u, v)^{\rho(u, v)} \|n_q\|^2} + \frac{\lambda(u, v) f_u f_u^T}{D^*(u, v)^{\sigma(u, v)} \|f_u\| \|n_q\|}, \quad (26)$$

where I is a 2×2 identity matrix, n_q is a column vector, f_u is a tangential vector (the partial derivative of $f(u, v)$ on u) at

the point $Q(u, v)$. $\lambda(u, v)$ is a saturation function defining the orientation of tangential velocity; $\|\cdot\|$ is 2-norm of a vector or a matrix. Besides, $\rho(u, v)$ and $\sigma(u, v)$ are defined as the weight of n_q and t_q , respectively:

$$\begin{aligned} \rho(u, v) &= \rho_0 \exp\left(\frac{1}{D_0 - D^*(u, v)}\right), \\ \sigma(u, v) &= \sigma_0 \exp\left(\frac{1}{D_0 - D^*(u, v)}\right), \\ \lambda(u, v) &= \begin{cases} 1, & v_T^T f_u n_q^T v_T > \lambda_0, \\ \frac{v_T^T f_u n_q^T v_T}{\lambda_0}, & -\lambda_0 \leq v_T^T f_u n_q^T v_T \leq \lambda_0, \\ -1, & v_T^T f_u n_q^T v_T < -\lambda_0, \end{cases} \end{aligned} \quad (27)$$

where ρ_0 is the repulsive parameter, σ_0 is the tangential parameter, D_0 is the minimum permitted distance between the boundary of the robot, and λ_0 is a small positive threshold of the saturation function $\lambda(u, v)$. Then, the actual velocity v_d can be expressed as

$$v_d = P(u, v)v_T = v_T - \frac{n_q^T v_T}{D^*(u, v)^{\rho(u, v)} \|n_q\|^2} n_q + \frac{\lambda(u, v) n_q^T v_T}{D^*(u, v)^{\sigma(u, v)} \|f_u\| \|n_q\|} f_u. \quad (28)$$

It can be seen from equation (28) that v_d consists of three parts: v_T can be called the target velocity; $(n_q^T v_T / D^*(u, v)^{\rho(u, v)} \|n_q\|^2) n_q$ is taken as the repulsive velocity; $(\lambda(u, v) n_q^T v_T / D^*(u, v)^{\sigma(u, v)} \|f_u\| \|n_q\|) f_u$ can be called the tangential velocity. Similarly, the perturbation matrix $P(u, v)$ can be divided into three parts: attractive matrix I , repulsive matrix $(n_q n_q^T / D^*(u, v)^{\rho(u, v)} \|n_q\|^2)$, and tangential matrix $(\lambda(u, v) f_u f_u^T / D^*(u, v)^{\sigma(u, v)} \|f_u\| \|n_q\|)$. Therefore, we can readjust the shape of the path by changing parameters $\rho(u, v)$ and $\sigma(u, v)$. However, the perturbation matrix in this method can fully consider the shape of the obstacle and the

position of the goal-directed locomotion and can more objectively describe the influence of the obstacle on the planned path.

Theorem 1. *The planned path can avoid obstacle safely.*

Proof. Vectors n_q and f_u are perpendicular exactly, i.e., $n_q^T f_u = 0$. Suppose that the distance between the boundary of the robot is close to the minimum permitted distance D_0 ; that is, suppose that $D^*(u, v) = D_0 + \delta(u, v)$ and $\delta(u, v)$ is a monotonically decreasing function:

$$D^*(u, v)^{\rho(u, v)} = (D_0 + \delta(u, v))^{\rho_0 e^{-(1/\delta(u, v))}} = D_0^{\rho_0} e^{-(1/\delta(u, v))} \left\{ \left(1 + \frac{\delta(u, v)}{D_0} \right)^{(D_0/\delta(u, v))} \right\}^{(\delta(u, v)/D_0) \rho_0 e^{-(1/\delta(u, v))}}. \quad (29)$$

It can be inferred that $\delta(u, v) \rightarrow 0+$, then $D_0^{\rho_0} e^{-(1/\delta(u, v))} \rightarrow 1$, $(1 + (\delta(u, v)/D_0))^{(D_0/\delta(u, v))} \rightarrow e$, and $(\delta(u, v)/D_0) \rho_0 e^{-(1/\delta(u, v))} \rightarrow 0$; therefore, $D^*(u, v)^{\rho(u, v)} \rightarrow 1$. $P(u, v)$ can be simplified as

$$P(u, v) = I - \frac{n_q n_q^T}{\|n_q\|^2} + \frac{\lambda(u, v) f_u f_u^T}{\|f_u\| \|n_q\|}. \quad (30)$$

Therefore, we can infer

$$n_q^T v_d = n_q^T P(u, v) v_T = n_q^T \left(I - \frac{n_q n_q^T}{\|n_q\|^2} + \frac{\lambda(u, v) f_u n_q^T}{\|f_u\| \|n_q\|} \right) v_T = \left(n_q^T - n_q^T + \frac{\lambda(u, v) n_q^T f_u n_q^T}{\|f_u\| \|n_q\|} \right) v_T. \quad (31)$$

Because the equation $n_q^T f_u = 0$ means that $n_q^T v_d = 0$, so the path is outside of the minimum permitted distance and there is no collision. $\square$

Theorem 2. *The planned path can reach the destination eventually.*

Proof. Because the goal of the path planning is to make the robot reach the destination safely, therefore, the velocity of the robot should have a component in the direction of the target velocity; that is, velocity v_T and v_d should satisfy $v_T^T v_d \geq 0$, and the planned path will converge to the target point. Besides, $v_d \approx v_T$ should be satisfied if the robot near to the destination $\xi_T = (x_T, y_T, z_T)$:

$$v_T^T v_d = v_T^T P(u, v) v_T = \|v_T\|^2 \left(1 - \frac{\cos^2 \langle v_T, n_q \rangle}{D^*(u, v)^{\rho(u, v)}} \right) + \frac{\lambda(u, v) v_T^T f_u n_q^T v_T}{D^*(u, v)^{\sigma(u, v)} \|f_u\| \|n_q\|}, \quad (32)$$

where $\langle \cdot \rangle$ is the inner product of vectors, where $\langle v_T, n_q \rangle$ denotes the angle between v_T and n_q . It is obvious that when

the robot approaches to the destination, then $D^*(u, v) \rightarrow 0+$:

$$\lim_{D^*(u, v) \rightarrow 0+} D^*(u, v)^{\rho(u, v)} = \lim_{D^*(u, v) \rightarrow 0+} (D^*(u, v))^{\rho_0 \exp(1/D_0 - D^*(u, v))} = +\infty. \quad (33)$$

As $D^*(u, v)^{\rho(u, v)} \rightarrow +\infty$ and $\cos^2 \langle v_T, n_q \rangle \leq 1$ hold, it can be inferred that $(1 - (\cos^2 \langle v_T, n_q \rangle / D^*(u, v)^{\rho(u, v)})) \geq 0$. From equation (30), it can be inferred that $\lambda(u, v) v_T^T f_u n_q^T v_T \geq 0$. Therefore, $v_T^T v_d \geq 0$ holds. When the robot approaches to the destination, that is to say, $D^*(u, v)^{\rho(u, v)} \rightarrow +\infty$ and $D^*(u, v)^{\sigma(u, v)} \rightarrow +\infty$, then according to equation (29), it can be inferred that $P(u, v) \rightarrow I$. Therefore $v_d \approx v_T$ holds. $\square$

4.5. The Algorithm Steps of Goal-Directed Locomotion. In the process of goal-directed locomotion, the snake-shaped robot continuously interacts with the environment according to the current environment state and updates and modifies the learning results to guide the robot's action selection. Finally, find a set of action sequences that can maximize the reward, and complete the goal-directed locomotion. The pseudocode of this method is shown in Algorithm 1.

5. Simulation Examples

5.1. Build the 3D Environment Model. To guarantee real-time performance, parameters are first learned offline according to different initial states and then correspondingly adjusted online based on the robot's states. To verify the effectiveness of the method proposed in this paper, a 40×40 virtual environment is set up, which contains several obstacles of different shapes. Suppose that $\xi = (x, y, z)$ is robot position and obstacle center is $\xi_0 = (x_0, y_0, z_0)$, with axis lengths a, b, c , and index parameters d, e, f :

$$F(\xi) = \left(\frac{x - x_0}{a} \right)^{2d} + \left(\frac{y - y_0}{b} \right)^{2e} + \left(\frac{z - z_0}{c} \right)^{2f}. \quad (34)$$

Then, the parameters of the path-integral-based reinforcement learning algorithm are given in Table 1.

Suppose that there are eight obstacles in environment as shown in Table 2, and $\rho_0(\xi) = 1$ and $\sigma_0(\xi) = 1$.

5.2. Collision-Free Goal-Directed Locomotion that the Robot Does Not Need to Climb Obstacles. The method proposed in this paper adds prior information and has historical experience as decision support, so collisions are avoided during path planning. After 660 rounds of training, robots can successfully reach the target point. When there is no trap area, the robot does not need to climb obstacles and can avoid all obstacles smoothly and swiftly and reach the destination eventually. Suppose that the starting position of the robots is $[5, -3, 0]$ and the target position is $[39, 39, 0]$. The planned path is shown in Figure 3.

It can be shown from Figure 3 that when the snake-shaped robot starts from different starting points, the algorithm proposed in this paper can avoid all obstacles smoothly and swiftly and reach the destination eventually.

5.3. Compared with Ant Colony Algorithm. By using Taguchi's experimental design method and other intelligent optimization methods [29, 30], the ant colony algorithm is compared to the algorithms proposed in this paper. Suppose that the starting position of the on-water automatic rescue intelligent robot is $[0, 20, 0]$, and the target position is $[19, 2, 0]$. In the ant colony algorithm of this paper, the

FIGURE 1: Structural diagram of the snake-shaped robot.

FIGURE 2: The snake-shaped robot path planning learning framework.

```

Initialize:  $S_t, Q(s, a)$ 
repeat
  (for each episode) Initialize:  $S$ 
  While  $S_t$  is not  $S_T$ 
    If (Probability less than  $\epsilon$ )
      choose  $A_t = \max Q(S_t)$ 
    Else
      Random choose  $A_t$ 
    End
    If (There is no collision)
      Take action  $A_t$ , return  $R$  and  $S$ 
      Update  $Q(s, a)$ 
    repeat
       $S_t \leftarrow S_T$ 
    until  $S_t = S_T$ 
  until The task of goal-directed locomotion is finished
End
  
```

ALGORITHM 1: The algorithm steps of goal-directed locomotion.

number of ants is 100. After each iteration, the pheromone attenuation ratio is 0.5, and each time the ant passes a path, the pheromone increase ratio is 2. It can be seen from Figure 4 that when the 3D ant colony algorithm is used for fast path planning, with different iteration times, the number of ants, etc., the planned path is also different. Moreover, 3D ant colony algorithm generally only seeks the shortest path, so the planned path is not very smooth as shown in Figure 5.

It can be seen from Figures 4 and 5 that when the number of iterations is the same, the best individual fitness curve varies with the number of ants. In other words, ant colony algorithm relies on prior knowledge to a large extent, so it has its shortcomings in fast path planning. The method proposed in this paper considers prior information, uses historical experience as decision support, and introduces a climbing coefficient, so that obstacles can be successfully

TABLE 1: The parameters of the path-integral-based reinforcement learning algorithm.

Parameters	Attenuation	Learning efficiency	ε -Greedy exploration probability	Reward	Reward	Reward
	$\gamma = 0.85$	$\alpha = 0.15$	$\varepsilon = 0.05$	$R_1 = -0.5$	$R_2 = -0.5$	$R_3 = 1$

TABLE 2: The shape parameters of each obstacle.

Obstacle	Shape	a	b	c	d	e	f	Center of the obstacle
1	Sphere	5	5	5	1	1	1	(20, 20, 0)
2	Sphere	6	6	6	1	1	1	(30, -2, 0)
3	Sphere	5	5	5	1	1	1	(15, 30, 0)
4	Cylinder	5	5	15	1	1	10	(22, 7, 0)
5	Cylinder	4	4	20	1	1	10	(35, 15, 0)
6	Cone	6	6	9	1	1	0.3	(3, 27, 0)
7	Cone	5	5	19	1	1	0.3	(6, 3, 0)
8	Cone	6	6	21	1	1	0.3	(30, 36, 0)

FIGURE 3: Goal-directed locomotion that the robot does not need to climb obstacles from different starting positions.

overcome. After more than 900 rounds of training, the two robots reached the target point safely without collision. In the process of goal-directed locomotion, the saturation λ represents the climbing ability of the robots. Furthermore, different λ will lead to different planned paths and will cause different climbing difficulties and different climbing heights. Therefore, we need to consider how to choose an appropriate coefficient λ to plan the most effective path. Suppose that the starting position of the robot is $[-10, -10, 0]$, and the target position is $[39, 20, 0]$. The planned path is shown in Figure 6.

It can be seen from Figure 6 that the planned path according to different saturation λ and the saturation λ of the robot is $1/6$. Both paths can bypass obstacles, escape the trap area, and reach the destination.

5.4. The Situation that the Robot Encounters the Trap Area.

During the process of goal-directed locomotion, the robot will encounter the trap area. It is necessary to consider the distribution of obstacles and the climbing ability of the robot; otherwise, the robot will not be carried out timely and effectively. Suppose that there are six obstacles in environment as shown in Table 3, and $\rho_0(\xi) = 1$ and $\sigma_0(\xi) = 1$. Suppose that the starting position is $[-10, -10, 0]$, and the target position is $[39, 20, 0]$.

It can be seen from Figure 7 that the trap area is formed by two different obstacles, and the snake-shaped robot will fall into the trap area. Therefore, we need to consider the climbing ability of the robot during path planning in the 3D environment.

FIGURE 4: The change of best individual fitness.

FIGURE 5: Planned path based on the ant colony algorithm.

FIGURE 6: The planned path of the snake-shaped robot with climbing ability.

TABLE 3: The shape parameters of each obstacle.

Obstacle	Shape	a	b	c	d	e	f	Center of the obstacle
1	Sphere	5	5	5	1	1	1	(20, 20, 0)
2	Sphere	7	7	7	1	1	1	(30, -2, 0)
3	Cone	6	6	9	1	1	0.3	(33, 27, 0)
4	Cone	12	12	19	1	1	0.3	(3, 27, 0)
5	Cone	12	12	21	1	1	0.3	(6, 3, 0)
6	Cone	6	6	21	1	1	0.3	(30, 36, 0)

FIGURE 7: The robot falls into the trap area.

6. Conclusions

This paper proposes a collision-free goal-directed locomotion method based on path-integral reinforcement learning in a three-dimensional environment. First, a path-integral reinforcement learning model is established, which describes in detail how the robot can accumulate experience from historical data to optimize decision-making. In order to improve the learning efficiency of this method, some irrelevant environmental conditions are removed. The Q table is updated based on the prior information and provided to the robot as historical experience, which greatly improves the learning efficiency of the robot. The simulation results show that the planned paths can avoid all obstacles smoothly and swiftly and reach the destination eventually.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61304088), the Fundamental Research Funds for the Central Universities (no. 2013QNA37), the China Postdoctoral Science Foundation

(no. 2015M581886), and Jiangsu Province Postdoctoral Research Foundation (no. 1501080B).

References

- [1] J. Gray, "The mechanism of locomotion in snakes," *Journal of Experimental Biology*, vol. 23, no. 2, pp. 101–120, 1946.
- [2] S. Hirose, *Biologically Inspired Robots: Snake-like Locomotors and Manipulators*, Oxford University Press, Oxford, UK, 1993.
- [3] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A review on modelling, implementation, and control of snake robots," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 29–40, 2012.
- [4] Z. Lu, S. Ma, B. Li, and Y. Wang, "Serpentine locomotion of a snakelike robot controlled by cyclic inhibitory CPG model," 2005.
- [5] K. Inoue, T. Sumi, and S. Ma, "CPG-based control of a simulated snake-like robot adaptable to changing ground friction," 2007.
- [6] G. Miller, "Snake robots for search and rescue," in *Neurotechnology for Biomimetic Robots*, MIT Press, Cambridge, MA, USA, 2002.
- [7] A. F. Zhang, S. G. Ma, B. Li et al., "Adaptive controller design for underwater snake robot with unmatched uncertainties," *Science China Information Sciences*, vol. 59, no. 5, pp. 1–15, 2016.
- [8] M. Malayjerdi, "Akbarzadeh A Analytical modeling of a 3-D snake robot based on sidewinding locomotion," *International Journal of Dynamics and Control*, vol. 7, no. 1, pp. 83–93, 2017.
- [9] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, vol. 17, no. 7, pp. 1283–1299, 2013.

- [10] A. Evangelos, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 17, no. 11, pp. 3137–3181, 2010.
- [11] P. Liljebäck and K. Y. Pettersen, "Waypoint guidance control of snake robots," 2011.
- [12] P. Liljebäck, I. U. Haugstuen, and K. Y. Pettersen, "Path following control of planar snake robots using a cascaded approach," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 111–126, 2012.
- [13] Z. Cao, Q. Xiao, R. Huang, and M. Zhou, "Robust neuro-optimal control of underactuated snake robots with experience replay," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 208–217, 2018.
- [14] A. Xi, T. W. Mudiyansele, D. Tao, and C. Chen, "Balance control of a biped robot on a rotating platform based on efficient reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 938–951, 2019.
- [15] B. Hu, Z. Cao, and M. Zhou, "An efficient RRT-based framework for planning short and smooth wheeled robot motion under kinodynamic constraints," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3292–3302, 2021.
- [16] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1179–1189, 2020.
- [17] E. Rezapour and P. Liljebäck, "Path following control of a planar snake robot with an exponentially stabilizing joint control law," *IFAC Proceedings Volumes*, vol. 46, no. 10, pp. 28–35, 2013.
- [18] E. Hasanabadi and M. J. Mahjoob, "Trajectory tracking of a planar snake robot using camera feedback," in *Proceedings of the 2nd International Conference on Control, Instrumentation and Automation*, Piscataway, MA, USA, 2011.
- [19] F. Matsuno and H. Sato, "Trajectory tracking control of snake robots based on dynamic model," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Piscataway, MA, USA, 2005.
- [20] A. Mohammadi, E. Rezapour, M. Maggiore, and K. Y. Pettersen, "Maneuvering control of planar snake robots using virtual holonomic constraints," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 884–899, 2016.
- [21] R. Ariizumi, R. Takahashi, M. Tanaka et al., "Head-trajectory tracking control of a snake robot and its robustness under actuator failure," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 6, pp. 2589–2597, 2018.
- [22] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: a path integral approach," 2010.
- [23] F. Stulp and O. Sigaud, "Robot skill learning: from reinforcement learning to evolution strategies," *Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.
- [24] E. A. Theodorou, *Iterative Path Integral Stochastic Optimal Control: Theory and Applications to Motor control*, University of Southern California, Los Angeles, CA, USA, 2011.
- [25] H. J. Kappen, "An introduction to stochastic control theory, path integrals and reinforcement learning," *Journal of Behavioral Robotics*, vol. 887, no. 1, pp. 149–181, 2007.
- [26] J. Yong, "Relations among ODEs, PDEs, FSDs, BSDEs, and FBSDEs," *Journal of Behavioral Robotics*, vol. 8, pp. 2779–2784, 1998.
- [27] E. A. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Behavioral Robotics*, vol. 11, pp. 3137–3181, 2010.
- [28] P. Yao, H. Wang, and Z. Su, "UAV feasible path planning based on disturbed fluid and trajectory propagation," *Chinese Journal of Aeronautics*, vol. 28, no. 4, pp. 1163–1177, 2015.
- [29] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.
- [30] J. Wang and T. Kumbasar, "Parameter optimization of interval Type-2 fuzzy neural networks based on PSO and BBBC methods," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 247–257, 2019.