Back-Stepping Experience Replay With Application to Model-Free Reinforcement Learning for a Soft Snake Robot

Xinda Qi, *Graduate Student Member, IEEE*, Dong Chen, *Member, IEEE*, Zhaojian Li, *Senior Member, IEEE*, and Xiaobo Tan, *Fellow, IEEE*

***Abstract*—In this letter, we propose a novel technique, Back-stepping Experience Replay (BER), that is compatible with arbitrary off-policy reinforcement learning (RL) algorithms. BER aims to enhance learning efficiency in systems with approximate reversibility, reducing the need for complex reward shaping. The method constructs reversed trajectories using back-stepping transitions to reach random or fixed targets. Interpretable as a bi-directional approach, BER addresses inaccuracies in back-stepping transitions through a purification of the replay experience during learning. Given the intricate nature of soft robots and their complex interactions with environments, we present an application of BER in a model-free RL approach for the locomotion and navigation of a soft snake robot, which is capable of serpentine motion enabled by anisotropic friction between the body and ground. In addition, a dynamic simulator is developed to assess the effectiveness and efficiency of the BER algorithm, in which the robot demonstrates successful learning (reaching a 100% success rate) and adeptly reaches random targets, achieving an average speed 48% faster than that of the best baseline approach.**

***Index Terms*—Deep reinforcement learning, experience replay, soft robot, snake robot, locomotion, navigation.**

I. Introduction

As a promising decision-making approach, reinforcement learning (RL) has drawn increasing attention for its ability to solve complex control problems and achieve generalization in both virtual and physical tasks, as evidenced in various applications, such as chess games [1], quadrupedal locomotion [2], and autonomous driving [3]. Considering the inherent infinite degrees of freedom of soft robots and their complicated interactions with environments [4], RL approaches were adopted for the control of complex soft robotic systems, such as soft manipulators [5], [6] and wheeled snake robots [7].

Manuscript received 20 January 2024; accepted 22 June 2024. Date of publication 16 July 2024; date of current version 19 July 2024. This article was recommended for publication by Associate Editor H. Hauser and Editor C. Laschi upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant CNS 2125484 and Grant ECCS-2024649. *(Corresponding author: Dong Chen.)*

Xinda Qi, Dong Chen, and Xiaobo Tan are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: qixinda@msu.edu; chendon9@msu.edu; xbtan@msu.edu).

Zhaojian Li is with the Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: lizhaoj1@msu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3427550, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3427550

As a typical challenge for RL, especially in tasks where complicated behaviors are involved, the learning efficiency suffers from the relatively large search space and the inherent difficulties of the tasks, which usually requires delicate reward shaping [8] to guide the policy optimization and to constrain the learning directions or the behavior styles. The RL agents have to successfully reach their goals for efficient learning before getting lost in numerous inefficient failure trials. Multiple strategies were proposed to address the hard exploration challenge with sparse rewards, including improving the exploration techniques for more versatile trajectories from intrinsic motivations [9], [10], [11], [12], and exploiting the information acquired from the undesired trails [13], [14], [15].

Compatible with these techniques that might improve learning efficiency, the motivation of BER proposed for off-policy RL is the human ability to solve problems forward (from the beginning to goal) and backward (from the goal to the beginning) simultaneously, which is different from the standard model-free RL algorithms that mostly rely on forward exploration. For example, in proving a complicated mathematical equation, an effective method is to derive the equation from both sides where the information of both the left-hand side (beginning) and the right-hand side (goal) is utilized, to which the reasoning process and the mechanism of BER are similar.

In this paper, a BER algorithm is introduced that allows the RL agent to explore bidirectionally, which is compatible with arbitrary off-policy RL algorithms. It is applicable for systems with approximate reversibility and with fixed or random goal setups. After an evaluation of BER with a toy task, it is applied to the locomotion and navigation task of a soft snake robot. The developed algorithm is validated on a physics-based dynamic simulator with a computationally efficient serpentine locomotion model based on the system characteristic. Comprehensive experimental results demonstrate the effectiveness of the proposed RL framework with BER in learning the locomotion and navigation skills of the soft snake robot compared with other state-of-the-art benchmarks, indicating the potential of BER in general off-policy RL and robot control applications.

II. Back-Stepping Experience Replay

A. Background

1) Reinforcement Learning: A standard RL formalism is adopted where an agent (e.g. a robot) interacts with an environment and learns a policy according to the perceptions and rewards. In each episode, the system starts with an initial state

$\boldsymbol{s}_0$ with a distribution of $p(\boldsymbol{s}_0)$, and the agent observes a current state $\boldsymbol{s}_t \in \mathcal{S} \subseteq \mathcal{R}^n$ in the environment at time step t. Then, an action $\boldsymbol{a}_t \in \mathcal{A} \subseteq \mathcal{R}^m$ is generated to control the agent based on the current policy π and s_t. Afterward, the system evolves to a new state $\boldsymbol{s}_{t+1}$ based on the action and transition dynamics $p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and a reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$ is collected by the agent for the learning before the termination of the episode. During the training process, the RL agent learns an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ mapping states to actions that maximize the expected return. The return is defined as the accumulated discounted reward $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, where γ is a discount factor.

The state value function $V^\pi(\boldsymbol{s}_t) = \mathbb{E}(R_t|\boldsymbol{s}_t)$ represents the expected return starting from state $\boldsymbol{s}_t$ following the current policy π, and the action value function $Q^\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}(R_t|\boldsymbol{s}_t, \boldsymbol{a}_t)$ represents the expected return starting from the state $\boldsymbol{s}_t$ with an immediate action $\boldsymbol{a}_t$ by following the current policy π. All optimal policies π^* share the same optimal Q-function Q^*, according to the Bellman equation [16]:

$$Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{s}' \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)} \left[r\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}'\right) + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q^*\left(\boldsymbol{s}', \boldsymbol{a}'\right) \right] \tag{1}$$

2) Deep Q-Networks (DQN) and Deep Deterministic Policy Gradient (DDPG): DQN is a model-free, off-policy RL approach suitable for agents operating in discrete action spaces [16]. It typically employs a neural network Q to approximate the optimal Q-function Q^*, selecting optimal actions: $\boldsymbol{a}^* = \arg\max_{\boldsymbol{a} \in \mathcal{A}} Q(\boldsymbol{s}_t, \boldsymbol{a})$. Exploration is often facilitated by the ϵ-greedy algorithm. To stabilize training, a *replay buffer* stores transition data $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ and is used to optimize Q with a loss $\mathcal{L} = \mathbb{E}(Q(\boldsymbol{s}_t, \boldsymbol{a}_t) - y_t)$, where the target y_t is calculated by using a periodically updated *target network* Q_{targ}: $y_t = r_t + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q_{\text{targ}}(\boldsymbol{s}_{t+1}, \boldsymbol{a})$, and using transitions in the *replay buffer*.

DDPG [17] is an off-policy RL algorithm that simultaneously learns a Q-function and a policy. DDPG interweaves the learning process of an approximator to Q^*, with an approximator to select $\boldsymbol{a}^*$, offering a unique adaptation for continuous action scenarios.

B. Algorithm for BER

The above classical off-policy RL algorithms often face challenges with systems characterized by sparse rewards or challenging tasks with rewards hard to reshape. In such scenarios, RL agents rarely achieve informative standard forward explorations due to a low success rate in reaching goals in complex problems without precise guidance [13]. To address these challenges, we propose a novel Back-stepping Experience Replay (BER) algorithm for tasks with different goals (Algorithm 1), designed to enhance the learning efficiency of off-policy RL algorithms. This is achieved by incorporating exploration methods in both forward and backward directions.

The BER algorithm requires at least an approximate reversibility of the system. This means that from a standard transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$, a back-stepping transition $(\boldsymbol{s}_{t+1}, \widetilde{\boldsymbol{a}}_t, \boldsymbol{s}_t)$ can be constructed, which is similar to a real transition $(\boldsymbol{s}_{t+1}, \widetilde{\boldsymbol{a}}_t, \boldsymbol{s}_{b,t})$ in the environment, i.e., $\boldsymbol{s}_{b,t} \approx \boldsymbol{s}_t$. The action in the back-stepping transition is calculated as $\widetilde{\boldsymbol{a}}_t = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$, where function f is dependent on the environment. The approximate reversibility is evaluated by a small upper bound K for all transitions during back-stepping:

$$\|\boldsymbol{s}_{b,t} - \boldsymbol{s}_t\| \leq K \cdot \|\boldsymbol{s}_{t+1} - \boldsymbol{s}_t\|, K < 1 \tag{2}$$

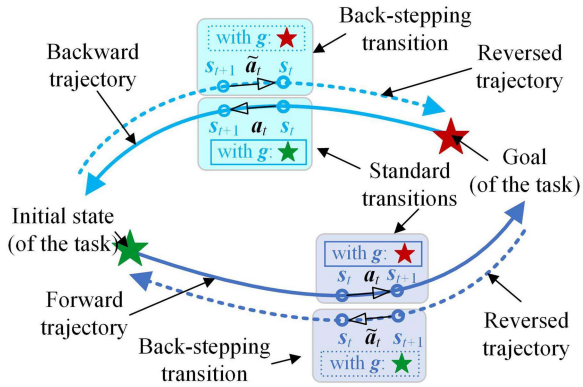

Fig. 1. Illustration of the Back-stepping Experience Replay, with navy and sky blue solid lines representing forward and backward explorations, respectively.

There exists a perfect reversibility when $K = 0$ with a probably complex function f, while an approximate reversibility might be achieved with a slightly larger K and a simpler and solvable function f. It is important to balance the accuracy and computational efficiency of f for effectively constructing back-stepping transitions that preserve enough information for the learning.

The idea of BER is simple yet effective: instead of solely relying on forward explorations (navy blue solid line in Fig. 1) from initial states to goals, which depend heavily on the randomness of forward trajectories to reach these goals, RL agents also navigate backward from the goals to the initial states in the tasks (sky blue solid line in Fig. 1). The standard transitions are sampled from the standard forward and backward exploration trajectories (solid lines in Fig. 1), where the initial states of themselves are included. Then, the back-stepping transitions are calculated based on the standard transitions to constitute the reversed trajectories (dashed lines in Fig. 1), where the virtual goals are set to be the original initial state in their corresponding standard trajectories, such that the reversed trajectories are guaranteed to reach their virtual goals and contribute to the learning efficiency.

During the explorations, the standard and the back-stepping transitions are collected and stored in separate replay buffers for training. A strategy $\mathbb{S}_t$ is used to sample the transitions from the standard replay R_f with a probability $P_{t,f}$ and from the back-stepping replay R_b with a probability $P_{t,b}$, where $P_{t,f} + P_{t,b} = 1$. For a system with imperfect reversibility, $P_{t,b}$ gradually drops to zero to purify the transition set for training because of the inaccurate back-stepping transition. The details of BER are shown in Algorithm 1. It should be noticed that the operator $\odot$ between the states and the goals also indicates the modification of the sequential data (e.g., the history data) when the back-stepping transitions are constructed.

The BER accelerates the estimation of Q-functions of the RL agent by using the reversed successful trajectories to bootstrap the networks. One interpretation of BER is a bi-directional search method for standard off-policy RL approaches, with a higher convergence rate and learning efficiency. The purification strategy of the transitions for training needs to be carefully tuned (e.g., tuning the probabilities $P_{t,f}$, $P_{t,b}$) and might be combined with other exploration techniques, to reach an accurate policy

Algorithm 1: Back-Stepping Experience Replay (BER).

Given:
– An off-policy RL algorithm $\mathbb{A}$. ▷ e.g. DDPG
– A probability P_b triggering backward trial.
– A strategy $\mathbb{S}_t$ for sampling transitions in replays
Require:
– Approximate reversibility of the system

1 Initialize $\mathbb{A}$ ▷ e.g. initialize networks
2 Initialize replay buffers R_f and R_b
3 **for** *epoch* $= 1 \rightarrow M$ **do**
4 Sample a goal $\boldsymbol{g}$ with an initial state $\boldsymbol{s}_0$.
5 **Forward trial** starts
6 **for** $t = 0 \rightarrow T_{end} - 1$ **do**
7 Sample an action $\boldsymbol{a}_t$ using the policy of $\mathbb{A}$:
8 $\boldsymbol{a}_t \leftarrow \pi(\boldsymbol{s}_t \odot \boldsymbol{g})$ ▷ e.g. $\odot \rightarrow$ diff, concat
9 Execute action $\boldsymbol{a}_t$, observe new state $\boldsymbol{s}_{t+1}$
10 **end**
11 **for** $t = 0 \rightarrow T_{end} - 1$ **do**
12 $r_t := r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, \boldsymbol{g})$
13 Store transition $(\boldsymbol{s}_t \odot \boldsymbol{g}, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1} \odot \boldsymbol{g})$ in R_f ▷ standard experience replay
  Construct a back-stepping transition:
14 $r_{b,t} := r(\boldsymbol{s}_{t+1}, \widetilde{\boldsymbol{a}}_t, \boldsymbol{s}_t, \boldsymbol{s}_0)$
15 Store transition $(\boldsymbol{s}_{t+1} \odot \boldsymbol{s}_0, \widetilde{\boldsymbol{a}}_t, r_{b,t}, \boldsymbol{s}_t \odot \boldsymbol{s}_0)$ in R_b ▷ BER
16 **end**
17 **Forward trial** ends
18 **Backward trial** starts with P_b
19 Swap the goal $\boldsymbol{g}$ and the initial state $\boldsymbol{s}_0$: $\boldsymbol{s}_0, \boldsymbol{g} = \boldsymbol{g}, \boldsymbol{s}_0$
20 Repeat line 6 - line 16
21 **Backward trial** ends
22 **for** $t = 1 \rightarrow N$ **do**
23 Sample a mini-batch B from the replay buffers $\{R_f, R_b\}$ using $\mathbb{S}_t$
24 Perform one step of optimization using $\mathbb{A}$ and mini-batch B
25 **end**
26 **end**

learning in the end and avoid the limitations brought by the bi-directional search method, e.g., non-trivial sub-optimum.

In the practical learning tasks, the accuracy and the complexity of the function $f : \widetilde{\boldsymbol{a}}_t = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$, which calculates the actions $\widetilde{\boldsymbol{a}}_t$ in the back-stepping transitions $(\boldsymbol{s}_{t+1}, \widetilde{\boldsymbol{a}}_t, \boldsymbol{s}_t)$, need to be balanced. An accurate f yields better reversibility (with smaller K in Eq. (2)) with more accurate back-stepping transitions and brings less bias and noise, while f itself could be computationally expensive or even unsolvable. On the other hand, a moderate relaxation of the accuracy of f might boost the efficiency of the calculation of back-stepping transitions, when the larger bias and the noises brought by the approximate reversibility (with larger K) are managed by the purification mechanism in BER.

1) A Case Study of BER: To illustrate the effectiveness and generality of BER, a general binary bit flipping game [13] with n bits was considered as an environment for the RL agent, where the state was the bit value array $\boldsymbol{s} = \{s_i\}_{i=1}^n \in \mathcal{S}$, $s_i \in \{0, 1\}$, and the action was the index of the chosen bit $\boldsymbol{a} \in \{1, \ldots, n\} = \mathcal{A}$ that was flipped. It was noticed that the game was completely reversible and $\widetilde{\boldsymbol{a}}_t = f(\boldsymbol{a}_t) = \boldsymbol{a}_t$ for any time step and transition. The initial state $\boldsymbol{s}_0 \in \mathcal{S}$ and the goal $\boldsymbol{g} \in \mathcal{S}$ were sampled uniformly and randomly, with a sparse non-negative reward: $r_t(\boldsymbol{s}, \boldsymbol{a}) = -[\boldsymbol{s} \neq \boldsymbol{g}]$. The game is terminated once $\boldsymbol{s} = \boldsymbol{g}$.

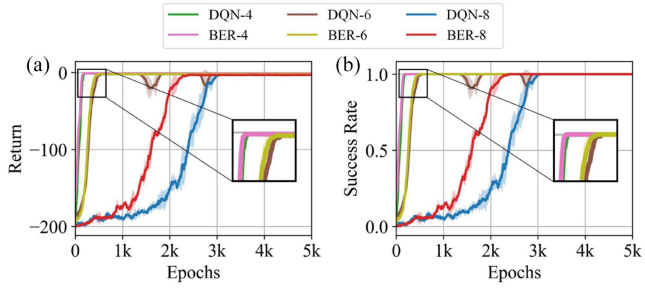

Fig. 2. Training experiments of the bit flip game with different algorithms and state dimensions. (a) Returns; (b) Success rates.

A simple ablation study was designed where a DQN and a DQN with BER were used for training when $n = 4, 6, 8$. The fully activated backward exploration and the use of back-stepping transitions were stopped after 1 k epochs directly. The experimental result (Fig. 2) showed that BER facilitated an effective and efficient policy learning for a general DQN approach, and contributed more when the problem became more complex (i.e., n was larger).

III. BER in Model-Free RL for a Soft Snake Robot

In this section, a locomotion and navigation task for a compact pneumatic soft snake robot with snake skins in our previous works [18], [19] is utilized to further evaluate the effectiveness and efficiency of BER with a model-free RL approach, where the robot learns both movement skills and efficient strategies to reach different challenging targets.

A. Soft Snake Robot and Serpentine Locomotion

Compared with soft snake robots where each air chamber was controlled independently [20], in this paper, a more compact soft snake robot with snake skins [18] is considered. There are only four independent air paths to generate the traveling-wave deformation of the robot, which enables the robot to traverse complex environments more easily by reducing the number of pneumatic tubing. The body of the robot consists of six bending actuators and each actuator is divided into four air chambers (Figs. 3(a), 3(d)) that connect to four air paths (Fig. 3(b)). Four sinusoidal waves with 90-degree phase differences and the same amplitude can be used as references of pressures in air paths to generate traveling-wave deformation (Fig. 3(c)), when the biases of waves induce unbalance actuation for steering of the robot.

Serpentine locomotion is adopted for the movement of the soft snake robot, where the anisotropic friction between the snake skins and the ground propels the robot during the traveling-wave deformation [21]. The artificial snake skins are designed with a soft substrate and embedded rigid scales (Fig. 3(e)); see [19] for more details.

To describe the serpentine locomotion of the robot, the dynamic model in [21] is adopted, where the body of the robot is modeled as an inextensible curve in a 2D plane with a total length L and a constant density ρ per unit length. The position of each point on the robot at time t is defined as:

$$\boldsymbol{X}(s, t) = (x(s, t), y(s, t)) \tag{3}$$

where s is the curve length measured from the tail of the robot.

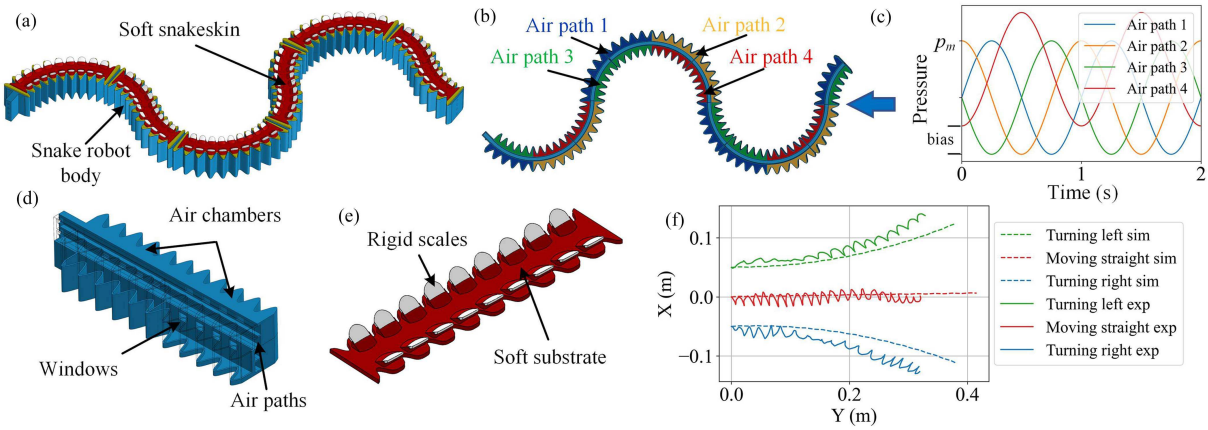

Fig. 3. The overview of the soft snake robot with skins. (a) The soft snake robot with soft snakeskins; (b) The connection between air chambers and air paths; (c) The actuation pressures for air paths; (d) The structure of one bending actuator; (e) The structure of soft snakeskin; (f) The simulation (sim) and experimental (exp) results of the trajectory of the COM of the snake robot on a rough paper surface.

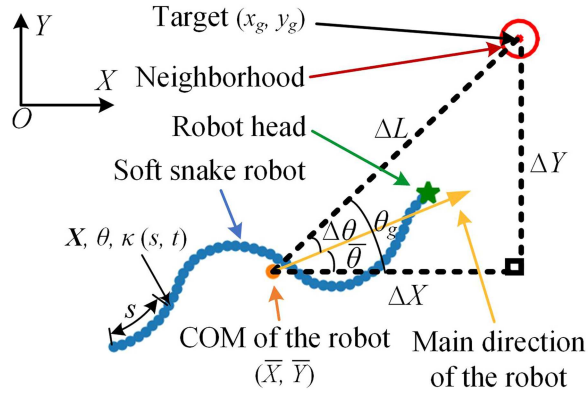

Fig. 4. The illustration of the soft snake robot with serpentine locomotion approaching a target.

By utilizing a mean-zero anti-derivative I_0 [22] ($I_0[f](s,t) = \int_0^s f(s',t)\mathrm{d}s' - \frac{1}{L}\int_0^L \mathrm{d}s \int_0^s \mathrm{d}s' f(s',t)$), the position $\boldsymbol{X}(s,t)$ and the orientation $\theta(s,t)$ (the angle between the local tangent direction and the X-axis of the inertial frame) of each point are described as a function of the position $\overline{\boldsymbol{X}}(t)$ and orientation $\overline{\theta}(t)$ (Fig. 4) of the center of mass (COM) of the robot:

$$\boldsymbol{X}(s,t) = \overline{\boldsymbol{X}}(t) + I_0[\boldsymbol{X}_s](s,t) \tag{4}$$

$$\theta(s,t) = \overline{\theta}(t) + I_0[\kappa](s,t) \tag{5}$$

where $\boldsymbol{X}_s = (\cos\theta, \sin\theta)$ and $\kappa(s,t)$ is the local curvature. $\overline{\boldsymbol{X}}(t) = \frac{1}{L}\int_0^L \boldsymbol{X}(s,t)\mathrm{d}s, \overline{\theta}(t) = \frac{1}{L}\int_0^L \theta(s,t)\mathrm{d}s$. The curvature $\kappa(s,t)$ is related to the local pneumatic pressure via:

$$\kappa(s,t) = K_b \cdot \Delta p(s,t) \tag{6}$$

where K_b is the proportional constant and $\Delta p(s,t)$ is the pressure difference between the two air chambers at point s.

The anisotropic friction $\boldsymbol{f}_{fric}$ between the snake skins and the ground is described as a weighted average of the independent components in different local directions (forward $\hat{\boldsymbol{f}}$, backward $\hat{\boldsymbol{b}}$, transverse $\hat{\boldsymbol{t}}$):

$$\begin{cases} \boldsymbol{f}_{fric} = -\rho g(\mu_t(\hat{\boldsymbol{u}}\cdot\hat{\boldsymbol{t}})\hat{\boldsymbol{t}} + \mu_l(\hat{\boldsymbol{u}}\cdot\hat{\boldsymbol{f}})\hat{\boldsymbol{f}}) \\ \mu_l = \mu_f H(\hat{\boldsymbol{u}}\cdot\hat{\boldsymbol{f}}) + \mu_b(1 - H(\hat{\boldsymbol{u}}\cdot\hat{\boldsymbol{f}})) \end{cases} \tag{7}$$

where $\hat{\boldsymbol{u}}$ represents the direction of the local velocity, μ_f, μ_b, and μ_t are the friction coefficients of the snakeskin in $\hat{\boldsymbol{f}}$, backward $\hat{\boldsymbol{b}}$, and $\hat{\boldsymbol{t}}$ directions, respectively. $H(x) = (1 + sgn(x))/2$, where sgn is the signum function.

The dynamics of each point of the snake robot is determined by Newton's second law:

$$\rho\ddot{\boldsymbol{X}} = \boldsymbol{f}_{fric} + \boldsymbol{f}_{inte} \tag{8}$$

where f_{inte} is the internal force in the robot body, which includes internal air pressure, bending elastic force, etc., with observations: $\int_0^L \boldsymbol{f}_{inte} = 0$ and $\int_0^L (\boldsymbol{X}(s,t) - \overline{\boldsymbol{X}}(t)) \times \boldsymbol{f}_{inte} = 0$.

Finally, the dynamics for the COM of the robot are derived using the equation (3)–(8) with the observations of f_{inte}; see [22] for more details.

Based on the dynamic model of the robot, which simplifies a dynamic system for all points of the robot to a single dynamic system for the COM of the robot, a simulator is designed with proper discretizations and numerical techniques for RL training. The simulation results matched the experimental results [19] of the soft snake robot when different pressure biases were applied for the robot's steering (Fig. 3(f)), where the wavy trajectories in the experiments were attributed to the limited number (25) of the tracking markers in the tests.

B. RL Formulation of Locomotion and Navigation of the Robot

In this paper, the locomotion and navigation of the soft snake robot are formulated as a Markov Decision Process (MDP) $\mathcal{M}$ and solved with a model-free RL. The $\mathcal{M}$ is defined as a tuple $\mathcal{M} = (\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, \gamma)$:

1) *Action space:* Compared with a random Central Pattern Generator (CPG) [7], more constrained sinusoidal waves are used to generate a smoother traveling-wave deformation of the robot for better locomotion efficiency. Besides, the learned controller of the robot is limited to avoid high-frequency pressure changes, i.e., the RL agent is only

able to generate an action to change the parameters of the waveform at the beginning of each actuation period $[0, T]$ that is same as the period of the sinusoidal waves, and one episode consists of multiple connected actuation periods. The sinusoidal pressure p_i for i-th channel of the robot is designed as:

$$p_i = p_m \sin\left(c \cdot \frac{2\pi}{T} t_r + \frac{(i-1)\cdot\pi}{2}\right) + b_{i,pre} + (b_i - b_{i,pre})\frac{t_r}{T} \tag{9}$$

where $t_r \in [0, T]$ is the relative time in one actuation period. p_m and $b_i \in [0, b_m]$ are the fixed magnitude and bias of the sinusoidal waves for the i-th channel, respectively, $i \in \{1, 2, 3, 4\}$. $b_{i,pre}$ is a one-step history of the wave bias b_i for the i-th channel with $b_{i,pre} = 0$ at the initial state. $c \in \{-1, 1\}$ is a variable to control the propagation direction of the traveling-wave deformation and thus can change the movement direction of the robot.

The action space $\mathcal{A}$ of the RL agent for locomotion and navigation of the robot is designed as:

$$\boldsymbol{a} = \{b_{a,1}, b_{a,2}, c\} \in \mathcal{A} \tag{10}$$

where b_i's are constructed by $b_{a,1} \in [-b_m, b_m]$ and $b_{a,2} \in [-b_m, b_m]$:

$$\begin{cases} b_1, b_3 = \max(0, b_{a,1}), -\min(0, b_{a,1}) \\ b_2, b_4 = \max(0, b_{a,2}), -\min(0, b_{a,2}) \end{cases} \tag{11}$$

At the beginning of each actuation period, based on the current policy, the RL agent observes the state and generates an action, which specifies the waveform of the pressures in that period to propel the snake robot. The wave design guarantees the continuity of the pressures across different actuation periods to avoid impractical sudden changes in the pressures and the robot's body shape.

2) *State space:* A goal-conditioned state is used for the learning of the RL agent for adapting to different random targets. Specifically, a relative representation of the snake robot's position and orientation with respect to the target is used as part of the state (Fig. 4):

$$\boldsymbol{s} = \{\Delta X, \Delta Y, \Delta\theta, \boldsymbol{b}_{a,1,pre}, \boldsymbol{b}_{a,2,pre}\} \in \mathcal{S} \tag{12}$$

where $\Delta X = x_g - \overline{X}$, $\Delta Y = y_g - \overline{Y}$ denote the relative position of the target to the COM of the snake robot, $\Delta\theta = \theta_g - \overline{\theta} \in (-\pi, \pi]$ represents the relative direction of the target to the main direction of the robot, and $\theta_g = \arctan(\Delta Y / \Delta X)$ is the angle between the line from the COM of the robot to the target and the X-axis, $\boldsymbol{b}_{a,1,pre}$ and $\boldsymbol{b}_{a,2,pre}$ are two-step histories of the action $b_{a,1}$ and $b_{a,2}$, respectively, with an initial setting of $\{0, 0\}$.

The velocities of COM of the robot are not included as part of the state because the value of the Froude number Fr [22] in serpentine locomotion of the snake robot is small, indicating that the frictional and gravitational effects dominate the inertial effect. Two-step histories (longer than one step) are introduced to compensate for the omission of the velocity state.

3) *Reward function:* The reward function r is pivotal for the RL agent to learn the desired behaviors. The training objective in this work is to drive the COM of the snake robot to reach a random target as soon as possible, with a preference for serpentine locomotion where the robot approaches the target along its main direction. Therefore, the reward assigned to the agent at time t is designed as:

$$r_t = \begin{cases} w_1 \frac{\Delta L_t}{\Delta L_0} + w_2 \frac{2\Delta\theta_{r,t}}{\pi} + R_g, & \Delta L_t \le \epsilon \\ w_1 \frac{\Delta L_t}{\Delta L_0} + w_2 \frac{2\Delta\theta_{r,t}}{\pi}, & \text{else} \end{cases} \tag{13}$$

where w_1 and w_2 are non-positive coefficients, R_g is a large sparse positive success reward once the COM of the robot enters a neighborhood of the target with a radius of ϵ. $\Delta L_t = \sqrt{\Delta X_t^2 + \Delta Y_t^2}$ is the distance between the COM of the robot and target at time t, and $\Delta L_t = \Delta L_0$ when $t = 0$. The deflection $\Delta\theta_{r,t} \in [0, \pi/2]$ is used in the reward to allow the robot to approach the target in a backward direction as well:

$$\Delta\theta_{r,t} = \begin{cases} |\Delta\theta_t|, & -\pi/2 \le \Delta\theta_t \le \pi/2 \\ \pi - |\Delta\theta_t|, & \text{else} \end{cases} \tag{14}$$

4) *Transition probabilities:* The transition probability, $\mathcal{T}(\boldsymbol{s}'|\boldsymbol{s}, a)$, characterizes the underlying dynamics of the robot system in the environment. In this study, we do not assume any detailed knowledge of this transition probability while developing our RL algorithm. However, it is noticed that some tests of the system are utilized in validating the function f to construct back-stepping transitions with acceptable reversibility, which distinguishes this approach from pure model-free approaches.

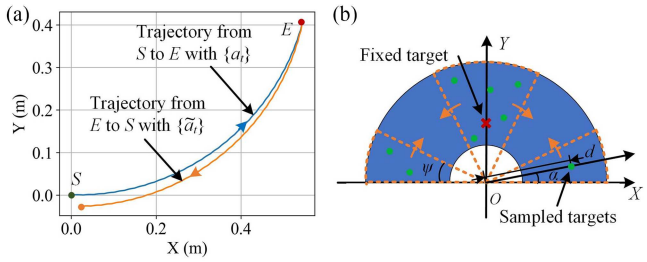

Fig. 5. (a). The approximate reversibility of the movement of the soft snake robot with snake skins. (b). The fixed target and the sampling range of the random targets.

C. Experiments of RL Algorithms

1) Experimental Setups: The RL experiments for the locomotion and navigation of the snake robot were conducted in a customized dynamic simulator which was developed based on the aforementioned serpentine locomotion model (Section III.A). The soft snake robot had a length of 0.5 m with a linear density of 1.08 kg/m. The frictional anisotropy between the snake skins and the ground was set as $\mu_f : \mu_b : \mu_t = 1 : 1 : 1.5$, and the maximum of the pressure bias b_m was set as the same as $p_m = 276$ kPa. The proportional constant K_b between the applied pressure difference and the curvature was set as 0.058 kPa·m. The period of the actuation and the sinusoidal waves was 1 s.

The serpentine locomotion of the soft snake robot demonstrated approximate reversibility (Fig. 5 A) in extensive simulations when the function f was designed as: $\tilde{\boldsymbol{a}}_t = f(\boldsymbol{a}_t) = \{b_{a,1}, b_{a,2}, -c\}$ when $\boldsymbol{a}_t = \{b_{a,1}, b_{a,2}, c\}$. The trajectories in extensive simulation results suggested a small $K < 1$ (in Eq. (2)) for locomotion and navigation of the soft snake robot when the above function f was used.

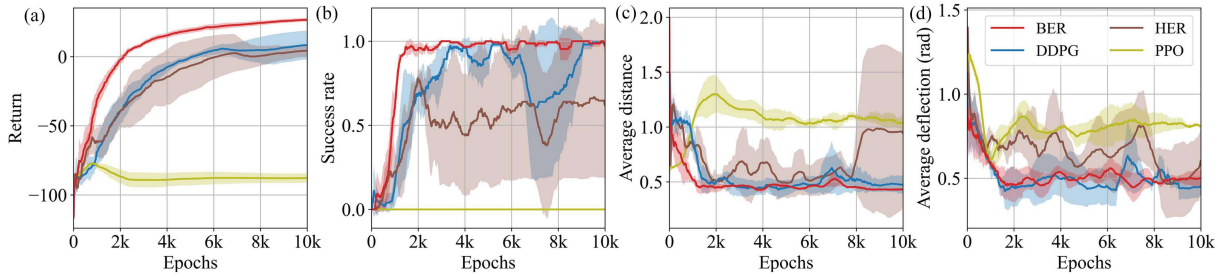

Fig. 6. Experimental results of the training for locomotion and navigation of the soft snake robot with one fixed target $(0, 0.5\text{ m})$. (a) Returns; (b) Success rates; (c) Average distances; (d) Average deflections.

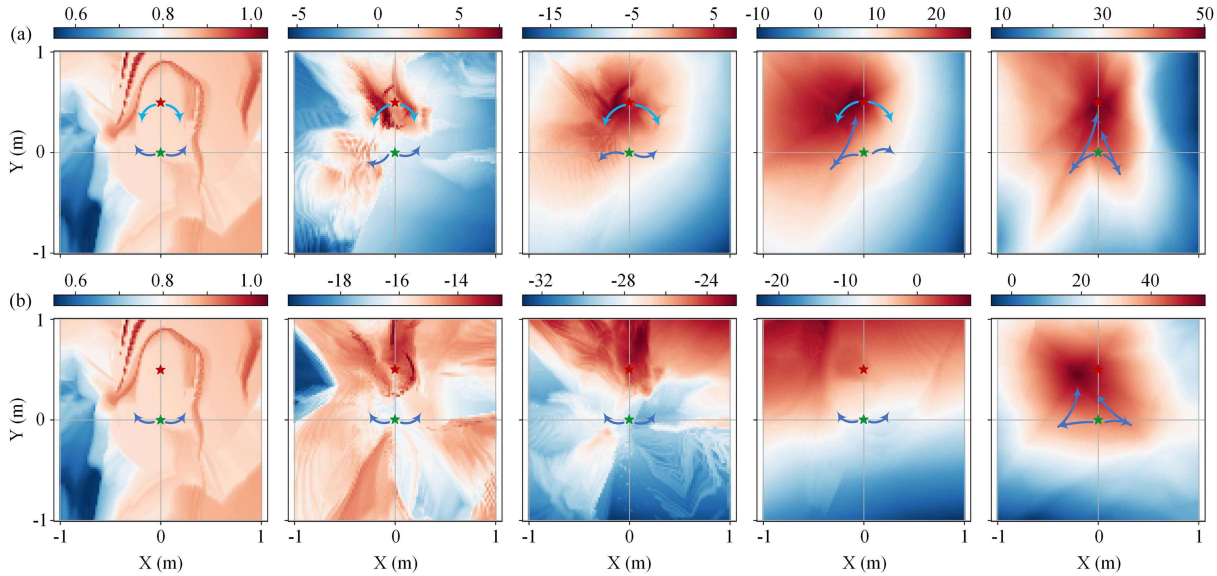

Fig. 7. Evolution of the maximum Q-value at different locations during the training (from left to right: initial state, epoch 100, 500, 1 k, 10 k), with blue arrows illustrating the directions of explorations. (a) Training with BER; (b) Training with DDPG.

The soft snake robot was initialized in the simulator by using a horizontal static curved shape ($(\overline{X}, \overline{Y}) = (0, 0), \overline{\theta} = 0$) with zero-value action histories and a target (with neighborhoods: $\epsilon = 0.03$ m), whose control policies were learned by using BER (with DDPG) and several state-of-art benchmark algorithms, including DDPG, HER [13], and PPO [23]. The number of total training epochs was 10,000 and the strategy to sample the transitions was $P_{t,b} = 0.5e^{-0.002i}$ when the index of epoch $i \leq 2500$, $P_{t,b} = 0$ when $i > 2500$, and $P_{t,f} = 1 - P_{t,b}$, $P_b = P_{t,b}$. The coefficients of the reward were selected as $\omega_1 = 0.15$, $\omega_2 = 1$ (while the choice of weights influences the learning performance, it was observed not to alter the general trend in performance comparison among the algorithms), and the termination condition for one episode was either the COM of the robot entering a neighborhood of the target and receiving a success reward ($R_g = 50$) or the exploration time exceeding 150 s.

The return, success rate, average distance (the averaged $\Delta L_t / L_0$ for each time step t), and average deflection (the averaged $\Delta\theta_{r,t}$ for each time step t) were used to evaluate the algorithms during the training, with moving-window averaging for training with different seeds ($l_{window} = 50$ epochs). Three training experiments with different random seeds (for parameter initialization) were conducted to evaluate each algorithm, where the solid line and the shaded area showed the mean and the standard deviation, respectively (Figs. 6 and 8). An AMD 9820X processor with 64 GB memory and Ubuntu 18.04 was used for the training.

2) Locomotion and Navigation With a Fixed Target: The performance of the algorithms was initially evaluated on the locomotion and navigation task of the robot, targeting a challenging fixed point $(x_g, y_g) = (0, 0.5\text{ m})$ (Fig. 5(b)). The experiment results of the training showed that both DDPG and BER were able to solve the task and learn policies to reach the fixed target successfully, while HER had worse stability and PPO was unable to solve the task within the epoch limitation (Fig. 6). It was also shown that BER had a faster convergence rate and better stability compared with other baseline algorithms.

The evolution of the maximum Q-value at different locations for the algorithms during the training process (with the same seed) revealed the underlying mechanism and the advantage of BER (Fig. 7). It was shown that the effective Q-values in the training with BER were estimated from both the start and the target locations, expediting the successful explorations and the convergence of the estimation. The BER learned a more informative Q-value distribution after 500 epochs than that of the baseline DDPG after 1000 epochs. The final Q-value distribution of BER was also more accurate than that of the baseline DDPG,

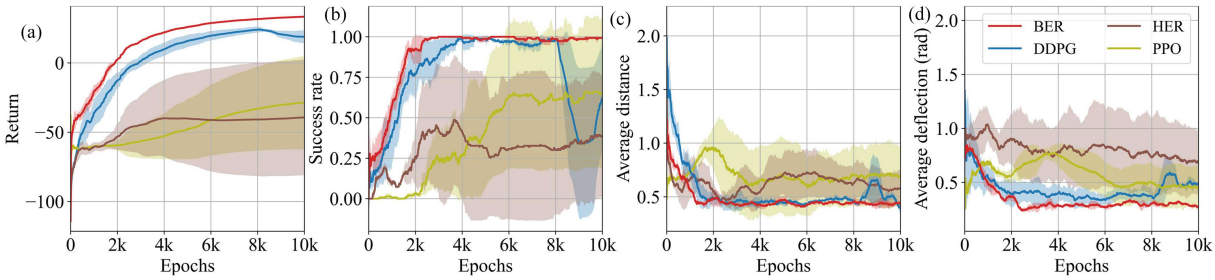

Fig. 8. Experimental results of the training for locomotion and navigation of the soft snake robot with random targets. (a) Returns; (b) Success rates; (c) Average distances; (d) Average deflections.

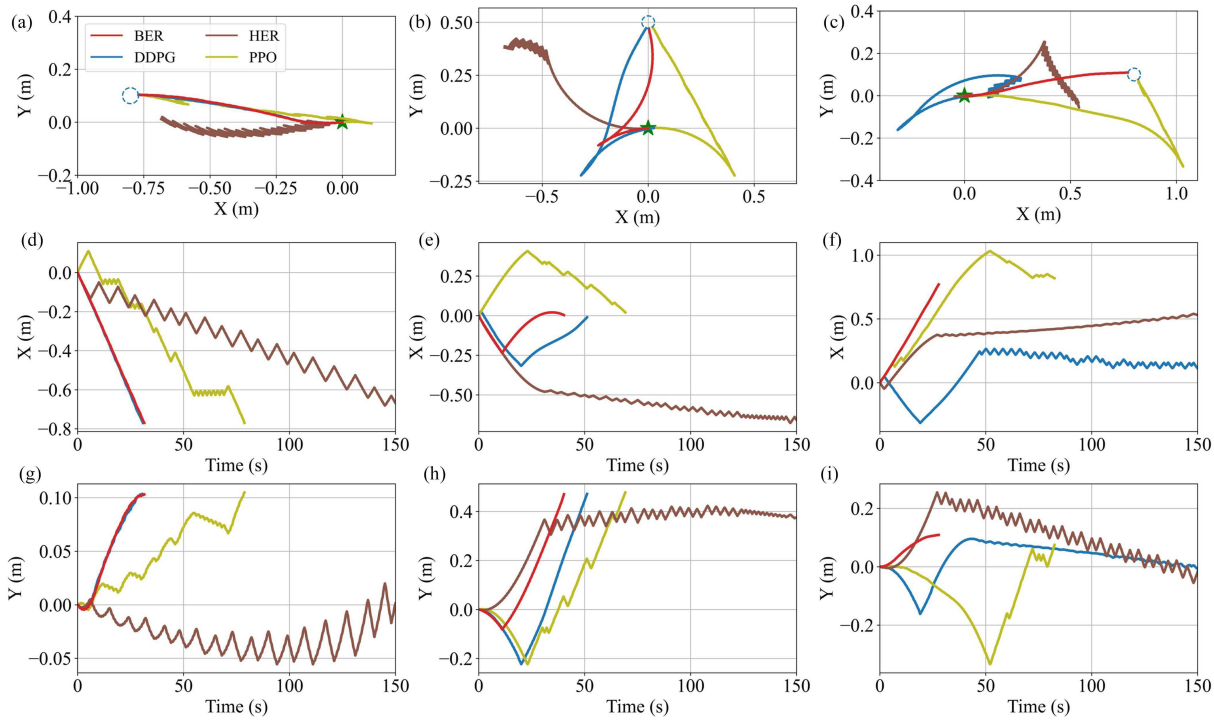

Fig. 9. Trajectories of the COM of the soft snake robot by using the controllers learned by different algorithms. (a) Trajectories with a backward target where the relationships between positions and time are shown in (d), (g); (b) Trajectories with a lateral target where the relationships between positions and time are shown in (e), (h); (c) Trajectories with a forward target where the relationships between positions and time are shown in (f), (i).

manifested by their shapes and the positions of the Q-value's peaks.

3) Locomotion and Navigation With Random Targets: A locomotion and navigation task of the soft snake robot with random targets was then explored by using different RL algorithms, where a half ring was used to randomly sample the target because of the system symmetry: $g \in \{(d, \alpha) \mid d \in [0.3, 1], \alpha \in [0, \pi]\}$ (Fig. 5(b)). Besides, a strategy was designed where the targets were sampled uniformly from gradually expanding areas for the i-th training epoch within the total n epochs: $g \in \{(d, \alpha) \mid d \in [0.3, 1], \alpha \in [0, \psi] \cup (\frac{\pi}{2} - \psi, \frac{\pi}{2} + \psi] \cup (\pi - \psi, \pi]\}$, $\psi = \frac{\pi}{4n^2} i^2$.

The training results revealed that BER outperformed all other tested benchmarks (Fig. 8). BER achieved the highest return and success rate during training, exhibiting more stable behavior and a smaller average deflection. In contrast, the baseline DDPG's performance declined when introduced to a variety of targets, despite its strong early-stage performance. HER struggled to learn to reach targets in different areas, indicating that the increasing of additional inefficient goals would not improve its performance but induce undesired behaviors, whereas PPO gradually learned an effective policy, a process that benefited from the random-goal training setup involving progressively changing targets.

The robot's trajectories further demonstrated BER's efficiency (Fig. 9), where controllers with median success rates from each algorithm were used for control. A video for these experiments in simulator can be viewed at https://youtu.be/Z0da6rVu9j8. Three representative targets were tested: $(-0.8$ m, 0.1 m) for moving backward, $(0, 0.5$ m) for moving towards a lateral target, $(0.8$ m, 0.1 m) for moving forward. The BER controller successfully and smoothly guided the robot to all targets. In contrast, the DDPG and HER controllers exhibited inefficient oscillations, possibly due to less accurate Q-function estimation. While the PPO controller managed to reach all targets, it also displayed oscillation and adopted a sub-optimal policy for the forward target $(0.8$ m, 0.1 m).

TABLE I
TESTING PERFORMANCE COMPARISONS OF DIFFERENT ALGORITHMS

Metrics	PPO	HER	DDPG	BER
Average velocity (m/s)	0.0061	0.0080	0.0114	**0.0169**
Average distance (m/m)	0.6241	0.5202	**0.3903**	0.4002
Average deflection (rad)	0.4049	0.6702	0.3915	**0.2920**
Success rate (%)	64.44	43.33	61.11	**100**

The quantitative results of the algorithms (Table I) were the average values tested by using the controllers trained with different seeds, and using 50 random targets sampled from the half-ring area (Fig. 5(b)). The average velocity ($v_{avg} = \Delta L_0/t_{ep}$, t_{ep}: episode length) indicated the efficiency of the learned controllers. Notably, the average velocity of the robot with the BER controller (0.0169 m/s) was approximately 48% faster than that of the DDPG baseline (0.0114 m/s), and significantly higher compared to other benchmarks. Besides, compared to other algorithms, BER not only learned an efficient controller based on the primary reward (highest average deflection: 0.2920 rd) but was also able to sacrifice the secondary reward to some extent (second highest average distance: 0.4002 m/m) for better performance. The success rate of BER reached 100% while the other baselines did not exceed 65%, which exhibited the advantage of BER in the locomotion and navigation learning of the soft snake robot.

IV. CONCLUSIONS AND DISCUSSIONS

A novel technique, Back-stepping Experience Replay, was proposed in this paper, which exploited the back-stepping transitions constructed by using the standard transitions in both forward and backward exploration trajectories, improving the learning efficiencies in off-policy RL algorithms for the approximate reversible systems. The BER was compatible with arbitrary off-policy RL algorithms, demonstrated by combining with DQN and DDPG in a bit-flip task and locomotion and navigation task for a soft snake robot, respectively.

A model-free RL framework was proposed for locomotion and navigation of a soft snake robot as an application of the proposed BER, where a conventional locomotion model for real snakes was adopted to describe the serpentine locomotion of the soft snake robot and to design a simulator for learning. An RL formulation for locomotion and navigation of the soft snake robot was built based on the characteristics of the robot. Extensive experiments showed that the proposed RL approach was able to learn an efficient controller that drove the soft snake robot approaching fixed or even random targets by using serpentine locomotion. For the tasks with random targets, the controller learned by using BER achieved a 100 % success rate and the robot's average speed was 48 % faster than that of the best baseline RL benchmark.

For future work, we will apply the proposed RL approach with BER to a physical soft snake robot system, to explore the simulation-to-reality gap and minimize such a gap using techniques like [24]. It is also noted that we did not consider obstacles in the environment in the current work. We plan to investigate extending the proposed approach to such cases. In addition, we will also study the influence of the function f and the approximate reversibility of general systems (e.g. robotic arms) on BER, and analyze the convergence properties of BER for proper state-of-the-art off-policy RL algorithms.

REFERENCES

[1] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] X. Liu, R. Gasoto, Z. Jiang, C. Onal, and J. Fu, "Learning to locomote with artificial neural-network and CPG-based control in a soft snake robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7758–7765.

[3] M. Hua et al., "Multi-agent reinforcement learning for connected and automated vehicles control: Recent advancements and future prospects," 2023, *arXiv:2312.11084*.

[4] C. Lee et al., "Soft robot review," *Int. J. Control, Automat. Syst.*, vol. 15, pp. 3–15, 2017.

[5] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 124–134, Feb. 2019.

[6] R. Jitosho, T. G. W. Lum, A. Okamura, and K. Liu, "Reinforcement learning enables real-time planning and control of agile maneuvers for soft robot arms," in *Proc. Conf. Robot Learn.*, 2023, pp. 1131–1153.

[7] X. Liu, C. D. Onal, and J. Fu, "Reinforcement learning of CPG-regulated locomotion controller for a soft snake robot," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3382–3401, Oct. 2023.

[8] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, vol. 99, pp. 278–287.

[9] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos, "Count-based exploration with neural density models," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2721–2730.

[10] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.

[11] L. Fox, L. Choshen, and Y. Loewenstein, "Dora the explorer: Directed outreaching reinforcement action-selection," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ry1arUgCW

[12] A. P. Badia et al., "Never give up: Learning directed exploration strategies," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Sye57xStvB

[13] M. Andrychowicz et al., "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html

[14] M. Fang, C. Zhou, B. Shi, B. Gong, J. Xu, and T. Zhang, "Dher: Hindsight experience replay for dynamic goals," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Byf5-30qFX

[15] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/c8d3a760ebab631565f8509d84b3b3f1-Abstract.html

[16] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[17] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[18] X. Qi, H. Shi, T. Pinto, and X. Tan, "A novel pneumatic soft snake robot using traveling-wave locomotion in constrained environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1610–1617, Apr. 2020.

[19] X. Qi, T. Gao, and X. Tan, "Bioinspired 3D-printed snakeskins enable effective serpentine locomotion of a soft robotic snake," *Soft Robot.*, vol. 10, no. 3, pp. 568–579, 2023.

[20] M. Luo, M. Agheli, and C. D. Onal, "Theoretical modeling and experimental analysis of a pressure-operated soft robotic snake," *Soft Robot.*, vol. 1, no. 2, pp. 136–146, 2014.

[21] D. L. Hu, J. Nirody, T. Scott, and M. J. Shelley, "The mechanics of slithering locomotion," *Proc. Nat. Acad. Sci.*, vol. 106, no. 25, pp. 10081–10085, 2009.

[22] D. L. Hu and M. Shelley, "Slithering Locomotion," in *Natural Locomotion in Fluids and on Surfaces: Swimming, Flying, and Sliding*. New York City, Ny, USA: Springer, 2012, pp. 117–135.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[24] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.