2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Learning to Locomote with Artificial Neural-Network and CPG-based Control in a Soft Snake Robot

Xuan Liu[1], Renato Gasoto[1,2], Ziyi Jiang[3], Cagdas Onal[1], Jie Fu[1]

***Abstract*— In this paper, we present a new locomotion control method for soft robot snakes. Inspired by biological snakes, our control architecture is composed of two key modules: A reinforcement learning (RL) module for achieving adaptive goal-tracking behaviors with changing goals, and a central pattern generator (CPG) system with Matsuoka oscillators for generating stable and diverse locomotion patterns. The two modules are interconnected into a closed-loop system: The RL module, analogizing the locomotion region located in the midbrain of vertebrate animals, regulates the input to the CPG system given state feedback from the robot. The output of the CPG system is then translated into pressure inputs to pneumatic actuators of the soft snake robot. Based on the fact that the oscillation frequency and wave amplitude of the Matsuoka oscillator can be independently controlled under different time scales, we further adapt the option-critic framework to improve the learning performance measured by optimality and data efficiency. The performance of the proposed controller is experimentally validated with both simulated and real soft snake robots.**

I. Introduction

Due to their flexible geometric shapes and deformable materials, soft continuum robots have great potentials in performing tasks under dangerous and cluttered environments [1]. However, planning and control of such type of robots remains a challenging problem, as these robots have infinitely many degrees of freedom in their body links, and soft actuators with hard-to-identify dynamics.

We are motivated to develop an intelligent control framework to achieve serpentine locomotion for goal tracking in a soft snake robot, designed and fabricated by [2]. The crucial observation from nature is that most animals with soft bodies and elastic actuators can learn and adapt to various new motion skills with only a few trials. These mechanisms have been studied for decades, and proposed as Central Pattern Generators (CPGs) or neural oscillators. As a special group of neural circuits located in the spinal cord of most animals, CPGs are able to generate rhythmic and non-rhythmic activities for organ contractions and body movements in animals. Such activities can be activated, modulated, and reset by neuronal signals mainly from two directions: bottom-up ascendant feedback information from afferent sensory neurons, or top-down descendant signals from high-level modules including mesencephalic locomotor region (MLR) [3] and motor cortex [4], [5].

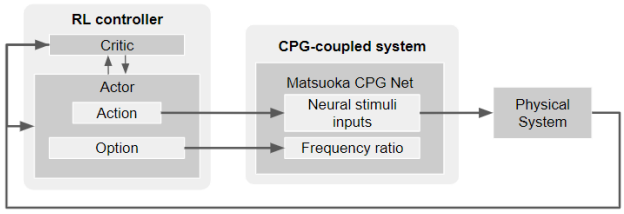


Fig. 1: Schematic view of learning based CPG controller.

In literature, bio-inspired control methods have been studied for the control design of locomotion control of rigid robots, including bipedal [6]–[9] and serpentine locomotion [10]–[15]. The general approach is to generate motion patterns mimicking animals' behaviors and then to track these trajectories with a closed-loop control design. In [3], the authors developed a trajectory generator for a rigid salamander robot using Kuramoto CPGs and used low-level PD controllers to track the desired motion trajectories generated by the oscillator. In [12], the authors improved the synchronization property of the CPG by adapting its frequency parameter with additional linear dynamics. In [14], the authors introduced a control loop that adjusts the frequency and amplitude of the oscillation for adapting to the changes of the terrain. Another recent work [15] employed Spiking Neural Net (SNN) under the regulation of Reward-Modulated Spike-Timing-Dependent Plasticity (R-STDP) to map visual information into wave parameters of a phase-amplitude CPG net, which generates desired oscillating patterns to locomote a rigid snake robot chasing a red ball.

Despite the success of bio-inspired control with rigid robots, it may be impossible to expect good performance if we apply the same control scheme to soft robots. This is mainly because that in these approaches, the trajectories generated by CPG require high-performance low-level controllers for tracking. The tracking performance cannot be reproduced with soft snake robots due to the nonlinear, delayed and stochastic dynamical responses from the soft actuators. In addition, most of these approaches only focus on improving locomotion performance under certain scenario, with a fixed goal position, or limited terrain types. To the best of our knowledge, none of the investigated work can track a randomly generated target with soft snake robot system.

To this end, we develop a bio-inspired intelligent controller for soft snake robots with two key components: To achieve intelligent and robust goal tracking with changing goals, we

[1]Xuan Liu is a PhD student under the supervision of Jie Fu and Cagdas Onal in the Robotics Engineering program at Worcester Polytechnic Institute. [2]Renato Gasoto is affiliated with NVIDIA. [3]Ziyi Jiang is an undergraduate student in the Electronic Engineering program at Xidian University.
xliu9, rggasoto, cdonal, jfu2@wpi.edu, jzyxbb@outlook.com

This work was supported in part by the National Science Foundation under grant #1728412, and by NVIDIA.

use model-free reinforcement learning [16], [17] to map the feedback of soft actuators and the goal location, into control commands of a CPG network. The CPG network consists of coupled Matsuoka oscillators [18]. It acts as a low-level motion controller to generate actuation inputs directly to the soft snake robots for achieving smooth and diverse motion patterns. The two networks form a variant of cascade control with only one outer-loop, as illustrated in Fig. 1.

To reduce the time and samples required for learning-based control, we leverages dynamic properties of Matsuoka oscillators in designing the interconnection between two networks. The reinforcement learning (RL) module learns to control *neural stimuli inputs* and *frequency ratio* to a CPG network given state feedback from the soft snake robot and the control objective. In analogy to autonomous driving, the neural stimuli inputs play the role of *steering control*, by modulate the amplitudes and phases of the outputs of the CPG net in real-time. The frequency ratio plays the role of *velocity control* as the changes the oscillating frequency of the CPG net result in changes in the locomotion velocity, measured at the head of the soft snake robot.

To conclude, the contributions of this work include:

- A novel bio-inspired tracking controller for soft snake robots, that combines *robust and optimality* in reinforcement learning and *stability and diversity in behavior patterns* in the CPG system.
- A detailed analysis of Matsuoka oscillators in relation to steering and velocity control of soft snake robots.
- Experimental validation in a real soft snake robot.

The paper is structured as follows: Section II provides an overview of the robotic system and the state space representation. Section III presents the design and configuration of the CPG network. Section IV discusses key properties of the CPG network and the relation to the design of artificial neural network for the RL module. Section V introduces curriculum and reward design for learning goal-reaching locomotion with a soft snake robot. Section VI presents the experimental validation and evaluation of the controller in both simulated and real snake robots.

II. System Overview

A full snake robot consists n pneumatically actuated soft links. Each soft link of the robot is made of Ecoflex™ 00-30 silicone rubber. The links are connected through rigid bodies enclosing the electronic components that are necessary to control the snake robot. In addition, the rigid body components have a pair of one directional wheels to model the anisotropic friction of real snakes. For each link with two chambers, only one chamber is active (pressurized) at a time.

The configuration of the robot is shown in Figure 2. At time t, state $h(t) \in \mathbb{R}^2$ is the planar Cartesian position of the snake head, $\rho_g(t) \in \mathbb{R}$ is the distance from $h(t)$ to the goal position, $d_g(t) \in \mathbb{R}$ is the distance travelled along the goal direction from the initial head position $h(0)$, $v(t) \in \mathbb{R}$ is the instantaneous planar velocity of the robot, and $v_g(t) \in \mathbb{R}$ is the projection of this velocity vector to the goal direction, $\theta_g(t)$ is the angular deviation between the goal direction and the velocity direction of the snake robot. According to [19], the bending curvature of each body link at time t is computed by $\kappa_i(t) = \frac{\delta_i(t)}{l_i(t)}$, for $i = 1, \ldots, 4$, where $\delta_i(t)$ and $l_i(t)$ are the relative bending angle and the length of the middle line of the i-th soft body link.

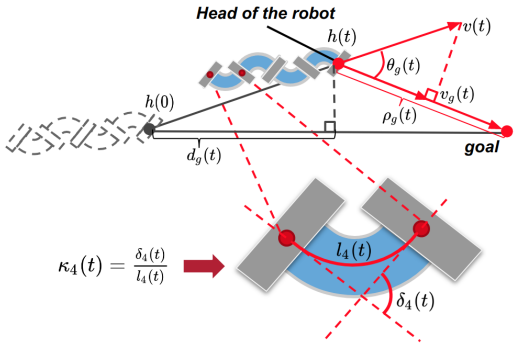


Fig. 2: Notation of the state space configuration of the robot.

In [20], we developed a physics-based simulator that models the inflation and deflation of the air chamber and the resulting deformation of the soft bodies with tetrahedral finite elements. The simulator runs in real-time using GPU. We use the simulator for learning the locomotion controller in the soft snake robot, and then apply the learned controller to the real robot.

III. Design of CPG Network for a Soft Snake Robot

In this section, we introduce our CPG network design consists of interconnected Matsuoka oscillators [21], [22].

Primitive Matsuoka CPG: A primitive Matsuoka CPG consists a pair of mutually inhibited neuron models. The dynanmical model of a primitive Matsuoka CPG is given as follows,

$$
\begin{aligned}
K_f \tau_r \dot{x}_i^e &= -x_i^e - a z_i^f - b y_i^e - \sum_{j=1}^{N} w_{ji} y_j^e + u_i^e \qquad (1)\\
K_f \tau_a \dot{y}_i^e &= z_i^e - y_i^e \\
K_f \tau_r \dot{x}_i^f &= -x_i^f - a z_i^e - b y_i^f - \sum_{j=1}^{N} w_{ji} y_j^f + u_i^f \\
K_f \tau_a \dot{y}_i^f &= z_i^f - y_i^f ,
\end{aligned}
$$

where the subscripts e and f represent variables related to extensor neuron and flexor neuron, respectively, the tuple (x_i^q, y_i^q), $q \in \{e, f\}$ represents the activation state and self-inhibitory state of i-th neuron respectively, $z_i^q = g(x_i^q) = \max(0, x_i^q)$ is the output of i-th neuron, $b \in \mathbb{R}$ is a weight parameter, u_i^e, u_i^f are the tonic inputs to the oscillator, and $K_f \in \mathbb{R}$ is the frequency ratio. The set of parameters in the system includes: the discharge rate $\tau_r \in \mathbb{R}$, the adaptation rate $\tau_a \in \mathbb{R}$, the mutual inhibition weights between flexor and extensor $a \in \mathbb{R}$ and the inhibition weight $w_{ji} \in \mathbb{R}$ representing the coupling strength with neighboring primitive oscillator. In our system, all coupled signals including x_i^q, y_i^q and z_i^q ($q \in \{e, f\}$) are inhibiting signals (negatively

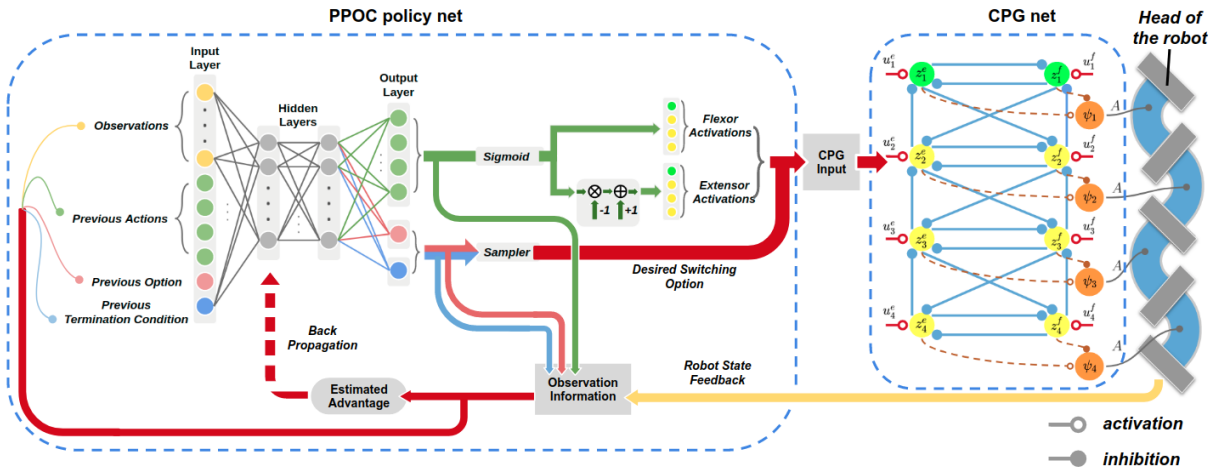


Fig. 3: Illustrating the input-output connection of the PPOC-CPG net.

weighted), and only the tonic inputs are activating signals (positively weighted).

Configurating the Matsuoka CPG network: The structure of the proposed CPG network is shown in Fig. 3. The network includes four linearly coupled primitive Matsuoka oscillators. It is an inverted, double-sized version of Network VIII introduced in Matsuoka's paper [21]. The network includes four pairs of nodes. Each pair of nodes (e.g., the two nodes colored green/yellow) in a row represents a primitive Matsuoka CPG (1). The edges are correspond to the coupling relations among the nodes. In this graph, all the edges with hollowed endpoints are positive activating signals, while the others with solid endpoints are negative inhibiting signals. The oscillators are numbered 1 to 4 from head to tail of the robot. In order to build the connection between the CPG network and robot actuators, we define the output of the i-th primitive Matsuoka CPG as

$$\psi_i = A(z_i^e - z_i^f), \tag{2}$$

where A represents the amplifying ratio of the difference between z_i^e and z_i^f of the i-th primitive oscillator. The CPG outputs $\boldsymbol{\psi} = [\psi_1, \psi_2, \psi_3, \psi_4]^T$ from the primitive oscillators are then normalized into the real region $[-1, 1]$ through $\hat{\psi}_i(t) = \psi_i(t)/\max_t |\psi_i(t)|$. We let $\hat{\psi}_i = 1$ for the full inflation of the i-th extensor actuator and zero inflation of the i-th flexor actuator, and vice versa for $\hat{\psi}_i = -1$. The actual pressure input to the i-th chamber is $\lambda_i \cdot \hat{\psi}_i$, where λ_i is the maximal pressure input of each actuator. The primitive oscillator with green nodes controls the oscillation of the head joint. This head oscillator also contributes as a rhythm initiator in the oscillating system, followed by the rest parts oscillating with different phase delays in sequence. Figure 3 shows all activating signals to the CPG network. For simplicity, we'll use a vector

$$\boldsymbol{u} = [u_1^e, u_1^f, u_2^e, u_2^f, u_3^e, u_3^f, u_4^e, u_4^f]^T \tag{3}$$

to represent all tonic inputs to the CPG net. To achieve stable and synchronized oscillations of the whole system, the following constraint must be satisfied [18]:

$$(\tau_a - \tau_r)^2 < 4\tau_r \tau_a b, \tag{4}$$

where $\tau_a, \tau_r, b > 0$. To satisfy this constraint, we can set the value of b much greater than both τ_r and τ_a, or make the absolute difference $|\tau_r - \tau_a|$ sufficiently small.

To determine the hyper-parameters in the CPG network that generate more efficient locomotion pattern, we employed a Genetic Programming (GP) algorithm similar to [23]. In this step, all tonic inputs are set as constant integer 1 for the simplicity of fitness evaluation.

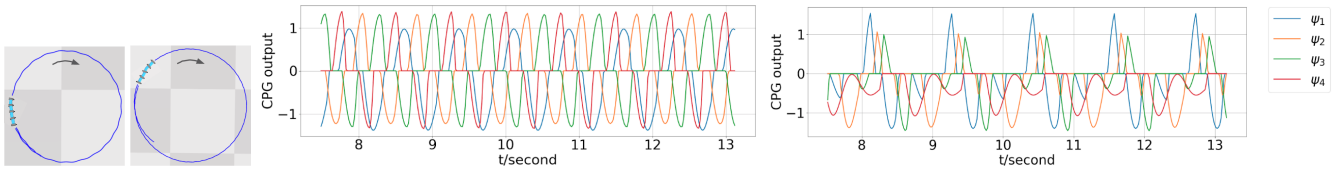
We define the fitness function–the optimization criteria–in GP as $F(t) = a_1|v_g(t)| - a_2|\theta_g(t)| + a_3|d_g(t)|)$, with a fixed goal always initiated on the heading direction of the snake robot, and all coefficients $a_1, a_2, a_3, T \in \mathbb{R}^+$ are constants[1]. This fitness function is a weighted sum over the robot's instantaneous velocity, angular deviation, and total traveled distance on a fixed straight line at termination time $t = T$. A better fitted configuration is supposed to maintain oscillating locomotion after a given period of time T, with faster locomotion speed $|v_g(T)|$ along the original heading direction. In addition, the locomotion pattern is also required to have less angular deviation from the initial heading direction (with a small $|\theta_g(T)|$), and with overall a longer travelled distance along the initial heading direction ($|d_g(T)|$).

The desired parameter configuration found by GP is given by Table. I in the Appendix.

IV. Design of Learning Based Controller with Matsuoka CPG Network

With constant tonic inputs, the designed Matsuoka CPG net can generate stable symmetric oscillations to efficiently drive the soft snake robot slithering straight forward, it does not control the steering and velocity to achieve goal-reaching behaviors with potentially time-varying goals. Towards intelligent navigation control, we employ a model-free RL algorithm, proximal policy optimization (PPO) [17], as the 'midbrain' of the CPG network. The algorithm is to learn the optimal policy that takes state feedback from the robot and controls tonic inputs and frequency ratio of the CPG net to generate proper oscillating waveform for reaching a goal. We

[1]In experiments, the following parameters are used: $a_1 = 40.0$, $a_2 = 100.0$, $a_3 = 50.0$, and $T = 6.4$ sec.



(a) (b) (c) (d)

Fig. 4: (a) Locomotion trajectory with biased amplitudes of tonic inputs $\boldsymbol{u} = [0.4, 0.6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]^T$ (3) to the CPG net. (b) Locomotion trajectory with biased duty cycles of tonic inputs. And plots of CPG outputs corresponding to tonic inputs with (c) biased amplitudes and (d) biased duty cycles.

would like to reduce the data and computation required for the learning to converge by leveraging crucial features of the Matsuoka CPG. Next, we present our configuration of RL guided by dynamical properties of Matsuoka CPG net.

A. *Encoding tonic inputs for steering control*

Steering and velocity control are key to goal-directed locomotion. Existing methods realize steering by either directly adding a displacement [3] to the output of the CPG system, or using a secondary system such as a artificial neural network to manipulate the output from multiple CPG systems [6]. We show that the maneuverability of Matsuoka oscillator provides us a different approach–tuning tonic inputs to realize the desired wave properties for steering.

In this subsection, we show that two different combinations of tonic inputs are capable of generating imbalanced output trajectories, which result in steering of the robot. One way is to apply biased values to each pair of tonic inputs. Another way is to introduce different duty cycles between the actuation of extensor and flexor.

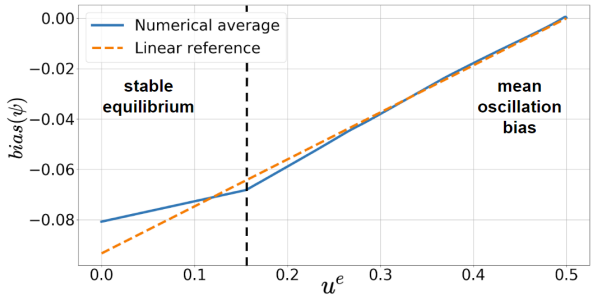


Fig. 5: Relation between oscillation bias and activation amplitude of u^e from a single primitive oscillator when $u^e + u^f = 1$.

Figure 5 shows the bias in the output ψ_i given the input u_i^e changing from 0 to 0.5 and $u_i^f = 1 - u_i^e$ in a primitive Matsuoka oscillator. Here, we approximate the bias of the steady-state output oscillation trajectory by taking the time-average of the trajectory [2]. From Fig. 5, we observe a linear mapping between tonic input u^e of a primitive oscillator and the bias of output, given $u^f + u^e = 1$.

In other words, the steering bias of a primitive Matsuoka oscillator is proportional to the amplitude of u^e when u^e and u^f are exclusive within $[0, 1]$. This key observation allows us to introduce a dimension reduction on the input space of the CPG net: Instead of controlling u_i^e, u_i^f for $i = 1, \ldots, n$ for n-link snake robot, we only need to control u_i^e for $i = 1, \ldots, n$ and let $u_i^f = 1 - u_i^e$. As the tonic inputs have to be positive in Matsuoka oscillators, we define a four dimensional action vector $\mathbf{a} = [a_1, a_2, a_3, a_4]^T \in \mathbb{R}^4$ and map $\mathbf{a}$ to tonic input vector $\mathbf{u}$ as follows,

$$u_i^e = \frac{1}{1 + e^{-a_i}}, \text{ and } u_i^f = 1 - u_i^e, \text{ for } i = 1, \ldots, 4. \quad (5)$$

This mapping bounds the tonic input within $[0, 1]$. The dimension reduction enables more efficient policy search in RL. Furthermore, different action vector $\mathbf{a}$ can be chosen to stabilize the system to limit cycles or equilibrium points.

Under the constraint of Eq. (5), steering can also be achieved by switching between different tonic input vectors periodically. When the duty cycle of actuating signals are different between flexors and extensors, an asymmetric oscillating pattern will occur. We construct two tonic input vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$, with $\boldsymbol{u_1} = [1, 0, 1, 0, 1, 0, 1, 0]^T$ and $\boldsymbol{u_2} = [0, 1, 0, 1, 0, 1, 0, 1]^T$. As Fig. 4d shows, when we set the duty cycle of $\boldsymbol{u_1}$ to be $1/12$ in one oscillating period, the rest $11/12$ of time slot for will be filled with $\boldsymbol{u_2}$. The CPG output on each link shows an imbalanced oscillation with longer time duration on the negative amplitude axis, indicating longer bending time on the flexor. As a result, the robot makes a clockwise (right) turn, with a circle trajectory presented in Fig. 4b.

We compare the steering control with two different methods in Fig. 4a and 4b. The corresponding trajectories in joint space are shown in Fig. 4c with biased amplitudes of tonic inputs and Fig. 4d with biased duty cycle of tonic inputs. In both experiments, the CPG outputs present noticeable biases to the negative amplitude axis. This indicates that all of the soft actuators are bending more to the right-hand side of the robot during the locomotion.

B. *Frequency modulation for velocity control*

As seen in Eq. (5), the constraint on each pair of tonic inputs prevents us from controlling locomotion speed with different amplitudes tonic inputs. We would like to determine another input to CPG net that controls the locomotion velocity.

Based on [22, Eq. (5), Eq. (6)], since K_f and $\mathbf{u}$ (Eq. (3)) are the control inputs, the mapping relations can be concluded as

$$\hat{\omega} \propto \frac{1}{\sqrt{K_f}}, \ \forall\ q \in \{e, f\}, i \in \{1, 2, 3, 4\}, \quad (6)$$

where $\hat{\omega}$ is the natural frequency of the oscillator.

[2]Based on Fourier series analysis, given a continuous real-valued P-periodic function $\psi_i(t)$, the constant term of its Fourier series has the form $\frac{1}{P}\int_P \psi_i(t)dt$.

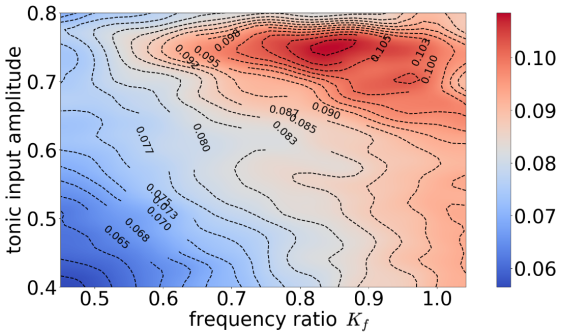


Fig. 6: Relating oscillating frequency and amplitude to the average linear velocity of serpentine locomotion.

Besides, the oscillation amplitude $\hat{A}$ is linearly proportional to the amplitude of tonic inputs, that is,

$$\hat{A} \propto A(u_i^q),\ \forall\ q \in \{e, f\}, i \in \{1, 2, 3, 4\}, \tag{7}$$

where $A(\cdot)$ is the amplitude function of a rhythmic signal.

Equations (6) and (7) show that the frequency and amplitude of the Matsuoka CPG system can be controlled *independently* by the frequency ratio K_f and the tonic inputs u_i^q, for $q \in \{e, f\}$. Figure 6 shows the distribution of locomotion velocity over different amplitudes and frequencies by taking 2500 uniform samples within the region $u_i^q \in [0.4, 0.8]$ for $q \in \{e, f\}, i \in \{1, 2, 3, 4\}$ with all u_i^q to be the same in one sample, and $K_f \in [0.45, 1.05]$. Noticed that in Fig. 6, with fixed tonic input, the average velocity increase nearly monotonically with the frequency ratio K_f. While the amplitude of tonic input does not affect the velocity that much, especially when K_f is low. Given this analysis, we use K_f to control the velocity of the robot.

It is noted that the frequency ratio K_f only influences the strength but not the direction of the vector field of the Matsuoka CPG system. Thus, manipulating K_f will not affect the stability of the whole CPG system.

C. The NN controller

We have now determined the encoded input vector of the CPG net to be vector $\mathbf{a}$ and frequency ratio K_f. This input vector of the CPG is the output vector of the NN controller. The input to the NN controller is the state feedback of the robot, given by $s = [\rho_g, \dot{\rho_g}, \theta_g, \dot{\theta_g}, \kappa_1, \kappa_2, \kappa_3, \kappa_4]^T \in \mathbb{R}^8$ (see Fig. 2). Next, we present the design of the NN controller.

We adopt a hierarchical reinforcement learning method called the option framework [24], [25] to learn the optimal controller regulating the tonic inputs (low-level primitive actions) and frequency ratio (high-level options) of the CPG net. The low-level primitive actions are computed at every time step. The high-level option changes infrequently as the robot needs not to change velocity very often for smooth locomotion. Specifically, each option is defined by $\langle \mathcal{I}, \pi_y : S \to \{y\} \times \mathbf{dom}(\mathbf{a}), \beta_y \rangle$ where $\mathcal{I} = S$ is a set of initial states. By letting $\mathcal{I} = S$, we allow the frequency ratio to be changed at any state in the system. Variable $y \in \mathbf{dom}(K_f)$ is a value of frequency ratio, and $\beta_y : S \to [0, 1]$ is the termination function such that $\beta_y(s)$ is the probability of changing from the current frequency ratio to another frequency ratio.

The options share the same Neural Network (NN) for their intro-option policies and the same NN for termination functions. However, these NNs for intro-option policies take different frequency ratios. The set of parameters to be learned by policy search include parameters for intra-option policy function approximation, parameters for termination function approximation, and parameters for high-level policy function approximation (for determining the next option/frequency ratio). Proximal Policy Optimization Option-Critics (PPOC) in the openAI Baselines [26] is employed as the policy search in the RL module.

Let's now review the control architecture in Figure 3. We have a Multi-layer perceptron (MLP) neural network with two hidden layers to approximate the optimal control policy that controls the inputs of the CPG net in Eq. (1). The output layer of MLP is composed of action $\mathbf{a}$ (green nodes), option in terms of frequency ratio (pink node) and the terminating probability (blue node) for that option. The input of NN consists of state vector (yellow nodes) and its output from the last time step. The purpose of this design is to let the actor network learn the unknown dynamics of the system by tracking the past actions in one or multiple steps [6], [27], [28]. Given the BIBO stability of the Matsuoka CPG net [18] and that of the soft snake robots, we ensure that the closed-loop robot system with the PPOC-CPG controller is BIBO stable. Combining with Eq. (5) that enforces limited range for all tonic inputs, the outputs of this control scheme are guaranteed to stay within a bounded region.

V. Learning-based Goal Tracking Control Design

In this section, we introduce the design of curriculum and reward function for learning goal-tracking behaviors with the proposed controller.

A. Task curriculum

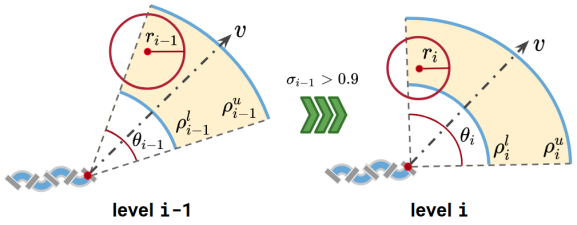


Fig. 7: Task difficulty upgrade from level $i-1$ to level i. As the curriculum level increases, goals are sampled at a narrower distance and wider angle, and acceptance area gets smaller.

Curriculum teaching [29] is used to accelerate motor skills learning under complex goal reaching task scenario. Through starting the trials with easy-to-reach goals, the agent will learn good policies more quickly. As the level increases, the robot improves the policy with more complex tasks. We design different levels of tasks as follows: At task level i, the center of goal is sampled from the 2D workspace based on the current location and head direction of the robot. For each

sampled goal, we say the robot reaches the goal if it is r_i distance away from the goal. The sampling distribution is a uniform distribution in the fan area determined by the range of angle θ_i and distance bound $[\rho_i^l, \rho_i^u]$ in polar coordinate given by the predefined curriculum.

As shown in Fig. 7, when the task level increases, we have $r_i < r_{i-1}$, $\theta_i > \theta_{i-1}$, $\rho_i^u > \rho_{i-1}^u$, and $\rho_i^u - \rho_i^l < \rho_{i-1}^u - \rho_{i-1}^l$; that is, the robot has to be closer to the goal in order to success and receive a reward, the goal is sampled in a range further from the initial position of the robot. We select discrete sets of $\{r_i\}, \{\theta_i\}$, $[\rho_i^l, \rho_i^u]$ and determine a curriculum table. We train the robot in simulation starting from level 0. The task level will be increased to level $i+1$ from level i if the controller reaches the desired success rate σ_i, for example, $\sigma_i = 0.9$ indicates at least 90 successful goal-reaching tasks out of $n = 100$ at level i.

B. Reward design

Based on the definition of goal-reaching tasks and their corresponding level setups, the reward is defined as

$$R(v_g, \theta_g) = c_v|v_g| + c_g \cos(\theta_g(t)) \sum_{k=0}^{i} \frac{1}{r_k} I(\rho_g(t) < r_k),$$

where $c_v, c_g \in \mathbb{R}^+$ are constant weights, v_g is the velocity towards the goal, θ_g is the angular deviation from the center of goal, r_k defines the goal range in task level k, for $k = 0, \ldots, i$, ρ_g is the linear distance between the head of the robot and the goal, and $I(\rho_g(t) < r_k)$ is an indicator function that outputs one if the robot head is within the goal range for task level k.

This reward trades off two objectives. The first term, weighted by c_v, encourages movement toward the goal. The second term, weighted by c_g, rewards the learner given the level of curriculum the learner has achieved for the goal-reaching task. For every task, if the robot enters the goal range in task level i, it will receive a summation of rewards $1/r_k$ for all $k \leq i$ (the closer to the goal the higher this summation), shaped by the approaching angle θ_g (the closer the angle to zero, the higher the reward).

If the robot agent reaches the goal defined by the current task level, a new goal is randomly sampled in the current or next level (if the current level is completed). There are two failing situations, where the desired goal will be re-sampled and updated. The first situation is starving, which happens when the robot stops moving for a certain amount of time. The second case is missing the goal, which happens when the robot keeps moving towards the wrong direction to the goal region for a certain amount of time.

VI. Experiment Validation

We used an artificial neural network configuration with 128×128 hidden layer neurons. At every step, the algorithm samples the current termination function to decide whether to terminate the current option and obtain a new frequency ratio K_f or keep the previous option. The backpropagation of the critic net was done with Adam Optimizer and a step size of $5e-4$. The starvation time for failing condition is 60 ms. The missing goal criterion is determined by whenever $v_g(t)$ stays negative for over 30 time steps.

A. Policy training

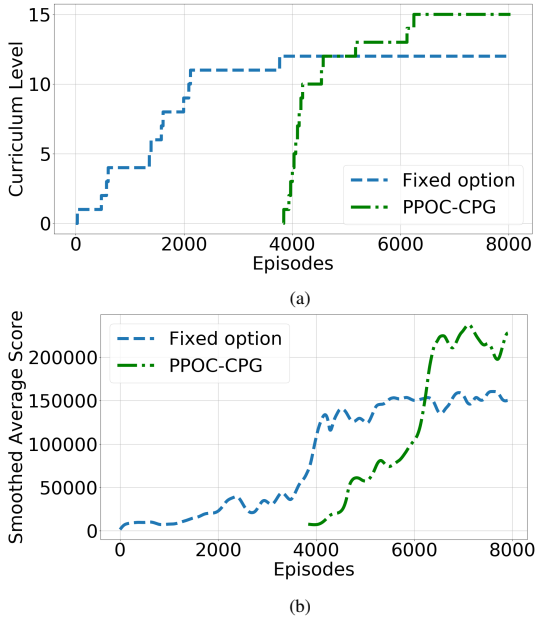


Fig. 8: (a) Learning progress of task level and (b) average learning score in the goal-reaching task.

Figure 8a shows improving performance over different levels versus the number of learning episodes. Figure 8b shows the corresponding total reward with respect to the number of learning episodes. As shown in Fig. 8a, we first train the policy net with fixed options (at this moment, the termination probability is always 0, and a fixed frequency ratio $K_f = 1.0$ is used). When both the task level and the reward cannot increase anymore (at about 3857 episodes), we allow the learning algorithm to change K_f along with termination function β, and keep training the policy until the highest level in the curriculum is passed. In this experiment, the learning algorithm equipped with stochastic gradient descent converges to a near-optimal policy after 6400 episodes of training. The whole process takes about 12 hours with 4 snakes training in parallel on a workstation equipped with an Intel Core i7 5820K, 32GB of RAM, and one NVIDIA GTX1080 ti GPU.

In order to compensate for simulation inaccuracies, most notably friction coefficients, we employed a domain randomization technique [30], in which a subset of physical parameters are sampled from a distribution with mean on the measured value. The Domain Randomization (DR) parameters used for training are on Table II, on the Appendix.

Figure 9 shows a sampled trajectory in the simulated snake robot controlled by the learned PPOC-CPG policy. Below the trajectory plot in Fig. 9 is the recorded pressure input trajectory to the first chamber. In the picture, green circles indicate the goals reached successfully, and the red circle represents a new goal to be reached next. The colors on the path show the reward of the snake state, with a color

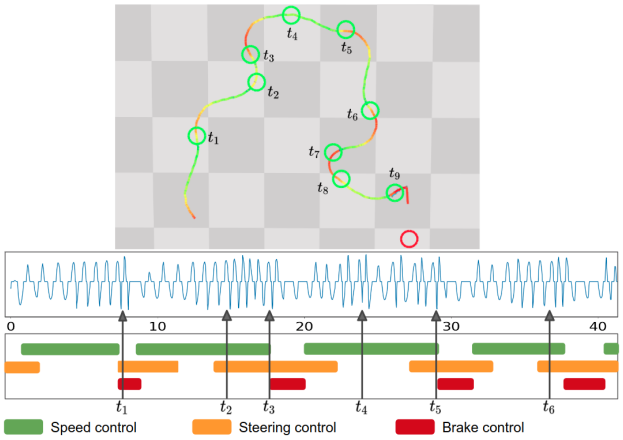


Fig. 9: A sample trajectory and the corresponding control events on the first link CPG output ψ_1.

gradient from red to green, indicating the reward value from low to high. Several maneuvering behaviors discussed in Section IV are exhibited by the policy. First, as the higher frequency can result in lower locomotion speed, the trained policy presents a specific two-phase behavior — (1) The robot starts from a low oscillation frequency to achieve a high speed when it is far from the goal; (2) then it switches to higher oscillation frequency using different options when it is getting closer to the goal. This allows it to stay close on the moving direction straight towards the goal. If this still does not work, the RL controller will use tonic inputs to force stopping the oscillation with the whole snake bending to the desired direction (see IV), and then restart the oscillation to acquire a larger turning angle.

B. Experiments with the real robot

We apply the learned policy directly to the real robot. The policy is tested on goal-reaching tasks guided by a mobile robot with an accuracy radius of $r = 0.175$ meter (The robot base has a 0.16 meter radius). The learning policy obtains the mobile robot position using the Mocap system in $60Hz$, and send the control commands to the robot through a low-latency wireless transmitter. A pair of example trajectories obtained from both simulation and real robot with identical policy are shown in Fig. 10a. The real robot tracks the goals with an average speed of 0.092m/s, while the simulation robot with tuned contact friction (see Table II in Appendix) reaches an average speed of 0.083m/s. Though trained on fixed goals only, the policy can also follow the slowly moving target in the test. From Fig. 10b and 10c, we notice that a delay of state evolution still exists between real robot and simulated robot, which is probably due to the inaccurate contact dynamics modeling by the physical engine. This could be fixed by either reducing the sim-to-real gap with more accurate modeling of the dynamics, or incorporating adaptive mechanism with velocity feedback to reduce the difference. We will leave this challenge as part of our future work.

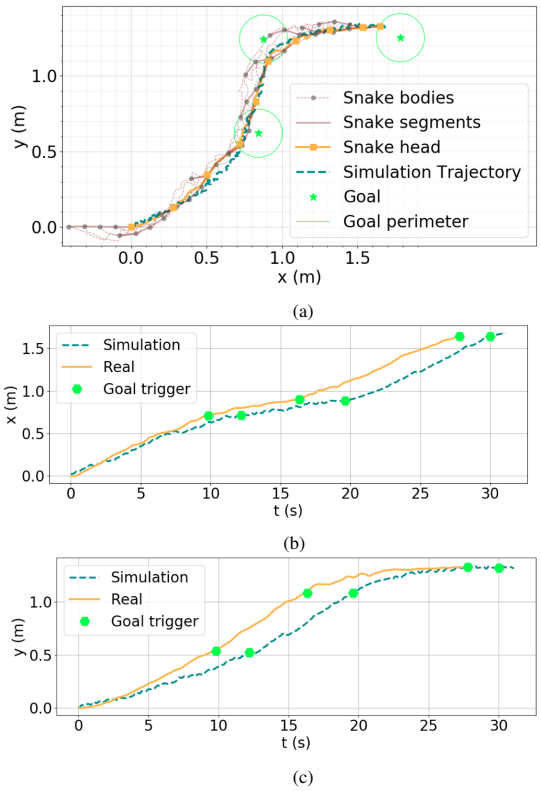


Fig. 10: Head trajectory of the simulation and real snake robot in consecutive goal-reaching tasks using PPOC-CPG controller trained in simulation in (a) x-y plane, (b) x-t plane and (c) y-t plane.

VII. Conclusion

The contribution of this paper is two folds: First, we investigate the properties of Matsuoka oscillator for learning diverse locomotion skills in a soft snake robot. Second, we construct a PPOC-CPG net that uses a CPG net to actuate the soft snake robot, and a reinforcement learning algorithm to learn a closed-loop near-optimal control policy that utilizes different oscillation patterns in the CPG net. This learning-based control method shows promising results on goal-reaching and tracking behaviors in soft snake robots. This control architecture may be extended to motion control of other robotic systems, including bipedal and soft manipulators. Our next step is to verify the scalability of the proposed control framework on the soft snake robot with more body segments and extend it to a three-dimensional soft snake robot and to realize more complex motion and force control in soft snake robots using distributed sensors and visual feedback.

References

[1] C. Majidi, "Soft robotics: A perspective–Current trends and prospects for the future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, 2014.

[2] C. D. Onal and D. Rus, "Autonomous undulatory serpentine locomotion utilizing body dynamics of a fluidic soft robot," *Bioinspiration & biomimetics*, vol. 8, no. 2, p. 026003, 2013.

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[4] A. Roberts, S. Soffe, E. Wolf, M. Yoshida, and F.-Y. Zhao, "Central circuits controlling locomotion in young frog tadpoles," *Annals of the New York Academy of Sciences*, vol. 860, no. 1, pp. 19–34, 1998.

[5] R. Yuste, J. N. MacLean, J. Smith, and A. Lansner, "The cortex as a central pattern generator," *Nature Reviews Neuroscience*, vol. 6, no. 6, p. 477, 2005.
[6] T. Mori, Y. Nakamura, M.-A. Sato, and S. Ishii, "Reinforcement learning for CPG-driven biped robot," in *AAAI*, vol. 4, 2004, pp. 623–630.
[7] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.
[8] J. Nassour, P. Hénaff, F. Benouezdou, and G. Cheng, "Multi-layered multi-pattern CPG for adaptive locomotion of humanoid robots," *Biological cybernetics*, vol. 108, no. 3, pp. 291–303, 2014.
[9] F. Dzeladini, N. Ait-Bouziad, and A. Ijspeert, "CPG-based control of humanoid robot locomotion," *Humanoid Robotics: A Reference*, pp. 1–35, 2018.
[10] A. Crespi, A. Badertscher, A. Guignard, and A. J. Ijspeert, "Swimming and crawling with an amphibious snake robot," pp. 3024–3028, 2005.
[11] A. Crespi and A. J. Ijspeert, "Online optimization of swimming and crawling in an amphibious snake robot," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 75–87, 2008.
[12] J.-K. Ryu, N. Y. Chong, B. J. You, and H. I. Christensen, "Locomotion of snake-like robots using adaptive neural oscillators," *Intelligent Service Robotics*, vol. 3, no. 1, p. 1, 2010.
[13] Z. Bing, L. Cheng, G. Chen, F. Röhrbein, K. Huang, and A. Knoll, "Towards autonomous locomotion: CPG-based control of smooth 3d slithering gait transition of a snake-like robot," *Bioinspiration & biomimetics*, vol. 12, no. 3, p. 035001, 2017.
[14] Z. Wang, Q. Gao, and H. Zhao, "CPG-inspired locomotion control for a snake robot basing on nonlinear oscillators," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 209–227, 2017.
[15] Z. Bing, Z. Jiang, L. Cheng, C. Cai, K. Huang, and A. Knoll, "End to end learning of a multi-layered SNN based on R-STDP for a target tracking snake-like robot," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 9645–9651.
[16] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.
[18] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation," *Biological cybernetics*, vol. 56, no. 5-6, pp. 367–376, 1985.
[19] M. Luo, M. Agheli, and C. D. Onal, "Theoretical modeling and experimental analysis of a pressure-operated soft robotic snake," *Soft Robotics*, vol. 1, no. 2, pp. 136–146, 2014.
[20] R. Gasoto, M. Macklin, X. Liu, Y. Sun, K. Erleben, C. Onal, and J. Fu, "A validated physical model for real-time simulation of soft robotic snakes," in *IEEE International Conference on Robotics and Automation*. IEEE, 2019.
[21] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biological cybernetics*, vol. 56, no. 5-6, pp. 345–353, 1987.
[22] ——, "Analysis of a neural oscillator," *Biological Cybernetics*, vol. 104, no. 4-5, pp. 297–304, 2011.
[23] A. J. Ijspeert, J. Hallam, and D. Willshaw, "Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology," *Adaptive behavior*, vol. 7, no. 2, pp. 151–172, 1999.
[24] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
[25] D. Precup, *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.
[26] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.
[27] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *International Conference on Robotics and Automation*. IEEE, 2018, pp. 1–8.
[28] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.
[29] A. Karpathy and M. Van De Panne, "Curriculum learning for motor skills," in *Canadian Conference on Artificial Intelligence*. Springer, 2012, pp. 325–330.
[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 23–30.

APPENDIX

TABLE I: Parameter Configuration of Matsuoka CPG Net Controller for the Soft Snake Robot.

Parameters	**Symbols**	**Values**
Amplitude	A	4.6062
$*$Self inhibition weight	b	**10.0355**
$*$Discharge rate	τ_r	**0.2888**
$*$Adaptation rate	τ_a	**0.6639**
Period ratio	K_f	1.0
Mutual inhibition weights	a_i	2.0935
Coupling weights	w_{ij}	8.8669
	w_{ji}	0.7844

TABLE II: Domain randomization parameters

Parameter	**Low**	**High**
Ground friction coefficient	0.1	1.5
Wheel friction coefficient	0.05	0.10
Rigid body mass (kg)	0.035	0.075
Tail mass (kg)	0.065	0.085
Head mass (kg)	0.075	0.125
Max link pressure (psi)	5	12
Gravity angle (rad)	-0.001	0.001