

# Energy-Efficient Slithering Gait Exploration for a Snake-like Robot based on Reinforcement Learning

Zhenshan Bing<sup>1</sup>, Christian Lemke<sup>2</sup>, Zhuangyi Jiang<sup>1</sup>, Kai Huang<sup>3</sup> and Alois Knoll<sup>1</sup>

<sup>1</sup>Department of Computer Science, Technical University of Munich, Germany

<sup>2</sup>Department of Computer Science, Ludwig Maximilian University of Munich, Germany

<sup>3</sup>School of Data and Computer Science, Sun Yat-sen University, China

bing@in.tum.de, Christian.Lemke@campus.lmu.de, jiangz@in.tum.de,  
huangk36@mail.sysu.edu.cn, knoll@in.tum.de

## Abstract

Similar to their counterparts in nature, the flexible bodies of snake-like robots enhance their movement capability and adaptability in diverse environments. However, this flexibility corresponds to a complex control task involving highly redundant degrees of freedom, where traditional model-based methods usually fail to propel the robots energy-efficiently. In this work, we present a novel approach for designing an energy-efficient slithering gait for a snake-like robot using a model-free reinforcement learning (RL) algorithm. Specifically, we present an RL-based controller for generating locomotion gaits at a wide range of velocities, which is trained using the proximal policy optimization (PPO) algorithm. Meanwhile, a traditional parameterized gait controller is presented and the parameter sets are optimized using the grid search and Bayesian optimization algorithms for the purposes of reasonable comparisons. Based on the analysis of the simulation results, we demonstrate that this RL-based controller exhibits very natural and adaptive movements, which are also substantially more energy-efficient than the gaits generated by the parameterized controller. Videos are shown at <https://videoviewsite.wixsite.com/rlsnake>.

## 1 Introduction

Snake-like robots, as a class of hyper-redundant mechanisms, carry the potential of being one kind of promising mobile robotic applications that are capable of traveling and performing tasks in diverse environments, such as disaster rescue, underwater exploration, and industrial inspection [Liljebäck *et al.*, 2012]. Since snake-like robots can only carry limited energy resources for field operations, it is important to develop energy-efficient gaits to reduce the impact of the power constraints. On one hand, optimizing the power consumption can prolong the service time of a robot and maximize its locomotion performance at the same time. On the other hand, a sufficient energy system may in return allow us to design a more lightweight robot or add other functional components [Tesch *et al.*, 2009]. However, it is challenging to design energy-efficient gaits for snake-like robots on the basis of their redundant degrees of freedom (DOF) and the complex interactions with the environment [Tucker, 1975].

Since the first snake-like robot was built in 1972, researchers have been working constantly on designing more advanced snake-like robots [Liljebäck *et al.*, 2012] and sophisticated gaits for robots with different types of mechanical configurations or terrains. Meanwhile, slithering gait has been considered as the most promising gait for snake-like robots to perform autonomous locomotion tasks, which imitates the serpentine locomotion of real snakes [Hu *et al.*, 2009]. Hirose first used the *serpenoid curve* to control a snake-like robot, which was an effective approach by imitating the real snake movement [Hirose, 1993]. Ma proposed another model *serpentine curve* to describe the locomotion of snakes by modeling their muscle characteristics and achieved a higher locomotive efficiency than the *serpenoid curve* by running simulations [Ma, 1999]. On the basis of these snake-like movement curves, the *gait equation*, as a robust and effective method, works as an abstract expression of gaits of a snake-like robot by describing joint angles as parameterized sinusoidal functions [Tesch *et al.*, 2009]. It allows for the emergence of complex behaviors from low-dimensional representations with only few key parameters, greatly expanding their maneuverability and simplifying user control. With this method, researchers developed several biological gaits for snake-like robots to move in the indoor and outdoor environment [Melo and Paez, 2014].

However, optimizing these parameterized gaits for the purpose of energy saving is difficult and limited, since they are confined to those abstracted gait parameters and only few studies have been reported.

Crespi *et al.* adopted a heuristic optimization algorithm to rapidly adjust the travel speed of the robot [Crespi and Ijspeert, 2008]. Tesch *et al.* used the Bayesian optimization approach to regulate those open-loop gait parameters for snake robots, which made the robot move faster and sturdier [Tesch *et al.*, 2011]. Gong *et al.* proposed a shape basis optimization algorithm to simplify the gait design parameter space and came up with a novel gait in granular materials [Gong *et al.*, 2016]. Even so, all these works still focus on optimizing the gait on the basis of the parameterized gait generation system, and have very limited effect on further improving gait efficiency.

This gait optimization task, however, corresponds to a complex control problem due to two primary reasons [Liljebäck *et al.*, 2012]. The extrinsic challenge comes from the complex dynamic interaction between the ground and the redundant mechanism with many degrees of freedom. Therefore, it is extremely important to model precisely and rapidly. The intrinsic challenge is how to synchronize and coordinate

all the body joints to exhibit a proper motion pattern integrally, which is expected to be both robust and efficient.

As an emerging technology, reinforcement learning (RL) reveals the nature of the learning process of locomotion in animals that can offer a model-free learning to master new skills or adapt to diverse environments. On this basis, this work offers a novel alternative to design the slithering gait of a snake-like robot based on reinforcement learning technology. Our main contributions are summarized as follows. First, we define the energy-efficiency metrics and introduce the parameterized slithering gait design method as the baseline, which is optimized by using a grid search method and the Bayesian optimization method in terms of energy efficiency. Second, we propose a gait controller using the state-of-the-art RL algorithm PPO in simulation. The learned gait exhibits surprising similarity to the natural movement of real snakes. Third, the results demonstrate that the learned gait successfully outperforms the parametrized slithering gait in terms of energy efficiency at a range of velocities.

## 2 Related Work

Unlike other land animals, snakes achieve diverse locomotion gaits by twisting their bodies on various terrain and exhibit undulatory locomotion. To imitate similar efficient movement, most snake-like robots are controlled by the kinematics-based method, which can be regarded as a process to simplify parametric representations of the snake-like trajectories [Hirose, 1993; Ma, 1999; Tesch *et al.*, 2009]. However, this method limits gait efficiency by only manually tuning those parameters, even without considering that it is time-consuming and inefficient.

Although there are few studies on optimizing the gaits for snake-like robots, studies on optimizing the locomotion gaits of other kinds of robots have been reported for the purpose of energy saving. Initially, researchers adapted techniques for multidimensional function optimization tasks to design efficient gaits, such as evolutionary algorithms [Chernova and Veloso, 2004] or policy gradient search algorithms [Kim and Uther, 2003; Kohl and Stone, 2004]. However, these algorithms are usually plagued by local optima, which makes the process slow, inefficient, and manually intensive. Afterwards, gait optimization methods based on prior knowledges are further investigated. Lizotte *et al.* first optimized the speed and smoothness of the gait with Gaussian process regression on a quadruped robot [Lizotte *et al.*, 2007]. Calandra *et al.* used Bayesian optimization approach to regulate those open-loop gait parameters for bipedal robots, which made the robot move faster and robuster [Calandra *et al.*, 2014]. Even though these optimized gaits outperform the hand-coded gaits, they are still very inefficient forms of locomotion compared with the natural movement achieved by animals.

As an intelligent trial-and-error learning method, RL brings new solutions for free gait generation tasks without knowing precise models or prior knowledge. Initially, RL was not widely used in the domain of robotics often presented with high-dimensional, continuous states and actions [Kober *et al.*, 2013]. However, with more and more advanced RL algorithms coming out, many robotic implementations are able to handle complicated tasks, such as gait generations [Peng *et al.*, 2017], dexterous manipulation [Rajeswaran *et al.*, 2017], and autonomous driving [Long *et al.*, 2018]. In [Cully *et al.*, 2015], two prototype experiments are shown in which RL can help robots recover from damage and adapt quickly like ani-

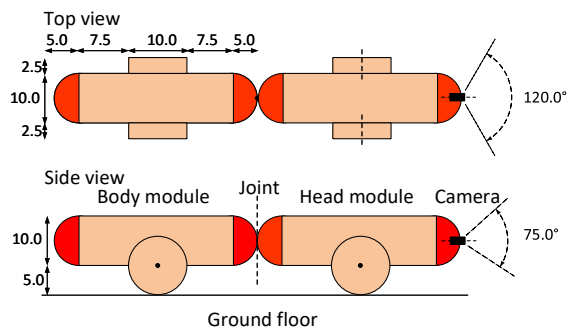


Figure 1: The top view and side view of the first two modules of the snake-like model (in centimetre).

mals do. A hexapod robot learns to walk fast and straight with broken or missing legs. And a robotic arm learns to reach its previous position goal with one or more stuck joints. Afterwards, RL technologies are increasingly used in free gait generation tasks. Schulman *et al.* [Schulman *et al.*, 2017] implemented their PPO algorithm on a collection of benchmark robotic locomotion tasks from 2D swimmer to 3D humanoid robot. Except for simply using RL-based methods to generate gaits for robots, they have been used to learning energy-efficient gaits. Kormushev *et al.* [Kormushev *et al.*, 2018] used RL to optimize the vertical center-of-mass trajectory and the walking pattern of a bipedal robot to exploit the passive compliance for the purpose of energy efficient walking. Yu *et al.* [Yu *et al.*, 2018] proposed an RL-based approach for creating low-energy and speed-appropriate locomotion gaits for four types of walking, including biped walking, quadruped galloping, hexapod walking, and humanoid running.

In this paper, we focus on using RL to learn slithering gaits for a snake-like robot with redundant degrees of freedom, so that the learned gaits can outperform those parameterized gaits in terms of energy efficiency.

## 3 Models and Metric Definition

In this section, we first introduce the snake-like robot model used for exploring different gaits. Then, we present our energy efficiency metric for comparing different gaits based on our robot model.

### 3.1 Snake-like Robot Model

We model and simulate our snake-like robot in MuJoCo [Todorov *et al.*, 2012], which is a physics engine offering fast and accurate robot simulation environment. The snake-like robot model used in this study is inspired by the ACM snake-like robot [Hirose, 1993], which uses eight joints and nine identical modules. The first module is used as the head module and equipped with a vision sensor for performing other tasks. Figure 1 shows the technical drawings of the first two modules of the snake-like robot model. The model acts in a dynamic environment, therefore its weight, friction on the ground, and actuator power are critical to enable availability. A uniform density of  $600 \text{ kg/m}^3$  is set for all components of the model. This density is selected based on the robot developed by Dowling [Dowling, 1996], which has about the same value including all mechanical and electrical components.

## Actuated Joints

The snake-like robot adopts actuated joints to bend its body to slither forward. All the joints are modeled as a servo motor and rotate along the perpendicular direction to the ground in a range of  $[-90^\circ, 90^\circ]$ . The force is limited to a range of  $[-20, 20]$  in Newton, which is adequate to propel the modules at a reasonable speed and strength. The maximum actuator torque can be calculated by multiplying the actuator force with the gear length of  $0.175\text{ m}$ , which is the half length of a module.

It is worth mentioning that a configuration with torque motor is also tested and works as well as the servo motors. For comparison purposes, the servo motor type is chosen because the *gait equation* controller also depends on this input type, which will be further explained in Section 4.

## Passive Damping Wheels

Each robot module is equipped with two passive wheels, which are used to imitate the anisotropic friction property of the snake skin. Like the movement of real snakes, those passive wheels enable a minimum friction in the direction of rotation and a high friction in the lateral direction. To enable the torque energy costs at all the joints, a constant damping property is added at each wheel joint. This constantly decelerates the rotation of the wheels, which imitates a low friction in forward direction. Otherwise, the robot will just form itself in a straight line and roll forward to gain distance while avoiding joint torque energy costs.

## 3.2 Energy Efficiency Metric

Mobile robots have to conserve their battery power so they can operate for long periods of time. Our goal is to design gaits that are capable to move in an energetic and economical manner and are still versatile to move at a range of velocities [Dowling, 1997].

### Power Profiles

The power profile is one set of the average power for each module of the robot during a test run, which offers detailed information to help improve the gait generations. For example, we can observe that which modules consume more power and therefore are more likely to heat up fast or broken.

For a snake-like robot with  $N$  joints, the averaged power  $\bar{P}_j$  for the  $j^{\text{th}}$  actuator is the averaged absolute value of the product of the torque  $\tau_j$  and its angular velocity  $\dot{\phi}_j$  during a run with  $k$  steps. Thus, our first metric can be expressed as

$$\bar{P}_j = \frac{1}{k} \sum_1^k \left| \tau_j \dot{\phi}_j \right|, \forall j \in [1, N]. \quad (1)$$

### Power Efficiency

Based on the power consumption of each module of the robot, the total power consumption  $P$  of all  $N$  actuators at each time step is calculated by

$$P = \sum_{j=1}^N \left| \tau_j \dot{\phi}_j \right|, \quad (2)$$

where the torque  $\tau_j$  is the product of its applied force  $f_j$  and its gear constant parameter  $h_j$  (the length of the actuator). The model uses actuators with a limited force of  $f_{max}$  as maximum force in both directions. With this property, the

normalized power consumption  $\hat{P}$  is calculated as

$$\hat{P} = \frac{1}{N} \sum_{j=1}^N \frac{\left| f_j h_j \dot{\phi}_j \right|}{f_{max} h_j \dot{\phi}_{max}}. \quad (3)$$

This normalized power consumption  $\hat{P}$  will be further used for reward definition.

With the variables of energy consumption  $P$  and the velocity  $v$ , several efficiency metrics can be calculated. A usual way is to calculate the Cost of Transport (COT), which is the power  $P$  or energy  $E$  divided by the mass  $m$ , the gravity  $g$  and the distance  $d$  or velocity  $v$ :

$$COT = \frac{E}{mgd} = \frac{P}{mgv} \quad (4)$$

This unit-less measure is also able to compare the efficiency between different mobile systems. Since the same robot model and the same environmental properties are compared, those constants only scale the results and would not provide any additional insight for the comparison. Thus, the second metrics is to calculate the Averaged Power per Velocity (APPV) as

$$APPV = \frac{P}{v} \quad (5)$$

Similar power efficiency metrics can be found in [Tesch *et al.*, 2009] and [Saito *et al.*, 2002].

## 4 Baseline Examples

This section provides two baseline examples, where the parameterized *gait equation* controller is presented to generate the slithering gait for our snake-like robot. By searching a grid of gait parameters with fixed intervals, we try to find out the best energy-efficiency gaits at different velocities that can be acquired by this controller. Then we use the Bayesian optimization algorithm to explore better parameter combinations in the range of searching grid, since the searching grid is relatively sparse.

### 4.1 Gait Equation Controller

The *gait equation* method represents a kinematic locomotion controller that uses a mathematical equation to describe its gait. In this work, an undulation gait equation extended from [Tesch *et al.*, 2009] is used for the purpose of comparison.

Params.	Descriptions	Values
$\omega$	Temporal frequency	0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0
$y$	Linear reduction	0.1, 0.2, 0.3, 0.4
$x$	Linear reduction	$(1 - y)$
$A$	Amplitude (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180
$\lambda$	Spatial frequency (in degrees)	40, 50, 60, 70, 80, 90, 100, 110, 120

Table 1: The gait parameters used for the grid search algorithm.

The *gait equation* controller is modeled as

$$\phi(n, t) = \left(\frac{n}{N}x + y\right) \times A \times \sin(\omega t + \lambda n). \quad (6)$$

$\phi(n, t)$  presents the joint angle value at time  $t$ , where  $n$  is the joint index and  $N$  is the joint amount.  $\lambda$  and  $\omega$  are the spatial and temporal frequency of the movement, respectively. The spatial frequency represents the cycle numbers of the wave and the temporal frequency represents the traveling speed of the wave.  $A$  is the serpentine amplitude and  $x$  and  $y$  are the constants for shaping the body curve.

To ensure a fair comparison, we use the grid search method to generate a variety of gaits and find out those parameter combinations with the best power efficiency at different velocities. The grid search method generates a Cartesian product from the parameters in Table 1, resulting in 6480 parameter sets. Then, each motion parameter set gets tested by running 1000 steps in the simulation environment. For each run, the first 200 time steps are ignored and the remaining 800 time steps are evaluated for collecting experiment data. This is done because it has been observed that the snake robot needs about 200 time steps to accelerate and then moves at a steady speed.

## 4.2 Bayesian Optimization Method

Since the searching grid is relative sparse, we further use the Bayesian optimization algorithm to search for better parameters that may be located in those grid intervals. This strategy has been used for optimizing snake-like robot gait parameters in [Tesch *et al.*, 2011] and other kinds of robots [Lizotte *et al.*, 2007; Calandra *et al.*, 2014].

The purpose of this method is to find a group of optimized parameters in (6) to achieve minimum power consumption at different velocities. The boundaries of these parameters are set as same as the value boundaries in Table 1. According to the *gait equation* controller, the maximum speed of the robot is mainly bounded by its temporal frequency  $\omega$ . Therefore, we perform the optimization process twelve times when  $\omega$  is uniformly sampled from  $[0, 3.0]$  at a step size of 0.25, which results in 12 trials. For each search, the exploration and exploitation samples are set as 10 and 100 for avoiding the local minimum.

## 4.3 Baseline Performance

The power consumption and the corresponding velocity results from the grid search algorithm and the Bayesian optimization algorithm are shown in Figure 2 as a point cloud of parameter sets using blue dot and orange diamond markers, respectively. The lowest points at different velocities in the point cloud have the highest efficiency. As we can see, the power requirement grows linearly with the increasing velocity. Most results from the Bayesian optimization controller (orange diamond marker) almost match the best-energy-efficiency points from the grid search method, which proves the applicability of the algorithm. And some points even exhibit better efficiency especially when the desired velocity is higher, which shows the capabilities of the Bayesian optimization algorithm for choosing better parameters for gait generation tasks. However, there are two outliers with low efficiency, which may be caused by falling into local minimum values during the optimization process.

In short, we demonstrate that the Bayesian optimization algorithm is a quick and effective method to find a proper parameter set for achieving desired velocity, especially when

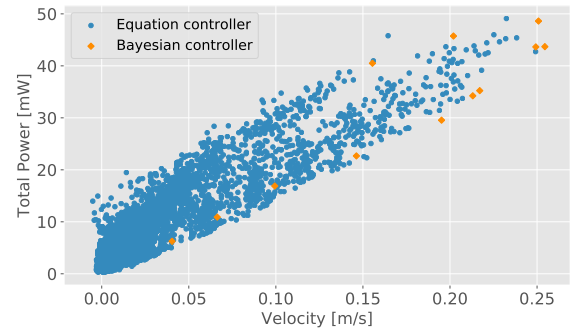


Figure 2: Baseline examples. This scatter plot shows the results from the grid search algorithm (blue point) and the Bayesian optimization algorithm (orange diamond), respectively. The coordinate of each data point represents the mean velocity and its corresponding power consumption.

most of the parameter sets in the searching space are distributed in the low velocity area ( $0 \text{ m/s} \sim 0.1 \text{ m/s}$ ).

## 5 Proposed RL-Based Controller

We begin this section by introducing the key ingredients of our reinforcement learning-based controller. Then the RL network architecture and the training configuration are introduced as well.

### 5.1 Reinforcement Learning Setup

Below we describe the details of the observation space, the action space, and the reward function.

#### Observation Space

The RL agent receives the environmental information via the observation space at each step. A proper choice of the observation space parameters is critical in RL, since the agent needs the right set of information to learn the causality of the given rewards based on its actions.

The observation space  $\mathbf{o}_i^t$  is given in Table 2. The joint position  $\phi_j$  and its angular joint velocity  $\dot{\phi}_j$  are required to learn the locomotion and represent the proprioceptive awareness of the robot. The head link velocity  $v_1$  helps to sense its global velocity, which offers better movement awareness. In order to learn a power efficient gait, the sense of energy consumption is necessary. Therefore, the actuator torque of each joint  $\tau_j$  is provided and can be interpreted in combination with  $\dot{\phi}_j$  to determine the total power usage. The specified target velocity  $v_t$  is passed to the environment and can be dynamically changed. This is required to control the velocity of the robot. In summary, an overall 26-DOF observation space is used in this work.

Symbols	Descriptions
$\phi_{1-8}$	Relative joint angular positions
$\dot{\phi}_{1-8}$	Relative joint angular velocity
$v_1$	Absolute head module velocity
$\tau_{1-8}$	Actuator torque output
$v_t$	Specified target velocity

Table 2: The observation space  $\mathbf{o}_i^t$  of the PPO controller.

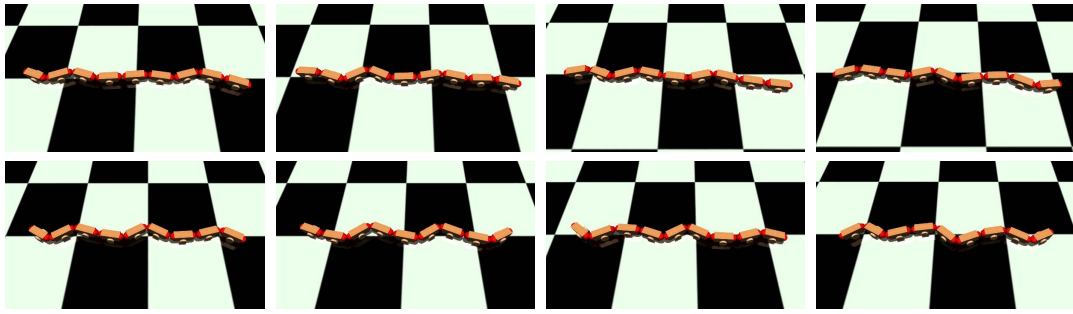


Figure 3: The montages of the snake-like robot model performing two learned gaits at different speeds. The frames are sorted in two rows from left to right and are recorded at intervals of 150 ms (the duration for 3 time steps). The first row is captured at a velocity of  $0.05 \text{ m/s}$  and appears to be more similar to the concertina gait. The second row is captured at a velocity of  $0.25 \text{ m/s}$  and resembles the slithering gait.

### Action Space

The action space  $\mathbf{a}_i^t$  of the environment has 8-DOF with finite continuous values in the range of  $[-1.5, 1.5]$ , which linearly translates to a corresponding joint angle  $\phi$  in range of  $[-90^\circ, 90^\circ]$ . Each action represents eight actuator angle positions of the servo motors. A uniform setup with servo motors is chosen so the environment between the two controllers can be compared. There has been no significant difference noticed between a servo and a torque setup.

### Reward Function

The objective of this experiment is to learn a power efficient locomotion for a variety of specified velocities. Therefore, the energy consumption and the difference between the actual model velocity and the target velocity are the main criterias to find a successful behavior. The challenge is to combine and weigh the variables into one numerical reward for each time step. Therefore, we first split the power efficiency and velocity criteria into two normalized reward function components.

First, a normalized reward is defined to maintain the specified velocities. The objective is to reach and maintain the target velocity  $v_t$  by comparing it with the head velocity  $v_1$ . The following function represents the velocity reward:

$$r_v = \left(1 - \frac{|v_t - v_1|}{a_1}\right)^{\frac{1}{a_2}} \quad (7)$$

The parameter  $a_1 = 0.2$  influences the spread of the reward curve, by defining the x-axis intersections with  $x = v_t \pm a_1$ .  $a_2 = 0.2$  affects the changes of the curves gradient. If  $|v_t - v_1| = 0.0$ , the velocity reward  $r_v$  has to be the maximum value 1.0.

Second, the normalized value of the total power usage  $\hat{P}$  in (3) is used to determine the power efficiency reward component  $r_P$ , which is represented by

$$r_P = r_{max} |1 - \hat{P}|^{b_1 - 2}. \quad (8)$$

Here,  $r_{max}$  controls the maximum reward value and  $b_1 = 0.6$  is the slope of the curve. The power efficiency is influenced by the desired target velocity. Therefore, the normalized value  $r_{max}$  represents this influence by limiting the maximum value of  $r_P$ .

Last, the rewards from the velocity  $r_v$  and the power efficiency  $r_P$  are combined to form the overall reward  $r$ :

$$r = \left(1 - \frac{|v_t - v|}{a_1}\right)^{\frac{1}{a_2}} |1 - \hat{P}|^{b_1 - 2} \quad (9)$$

This equation replaces the  $r_{max}$  in (8) with  $r_v$ . With that, the maximal power efficiency depends on the absolute value of the difference between the desired velocity and the robot velocity.

### 5.2 Network Architecture

Given the input (observation  $\mathbf{o}_i^t$ ) and the output (action  $\mathbf{a}_i^t$ ), we now elaborate the policy network mapping  $\mathbf{o}_i^t$  to  $\mathbf{a}_i^t$ . We design a fully connected 2-hidden-layer neural network as a non-linear function approximator to the policy  $\pi_\theta$ . The input layer has the same dimension as the action space  $\mathbf{o}_i^t$ . Both hidden layers have 200 ReLU units and the final layer outputs the joint position commands for the robot. In order to train the network, the PPO algorithm adapted from [Schulman *et al.*, 2017] is used for training it.

### 5.3 Training Configuration

To enable the learning of different velocities, the parameter  $v_t$  is changed by iterating over 0.05, 0.1, 0.15, 0.20, and 0.25 for each episode while training. Meanwhile, to simplify the beginning of the learning process the first 100 episodes are trained with a fixed target velocity of  $0.1 \text{ m/s}$ .

We train our policy network on a computer with an i7-7700 CPU and a Nvidia GTX 1080 GPU. Based on the learning curve a total of 3 million time steps (about 1400 updates) are used for training. With the environment settings of  $50 \text{ ms}$  per time step, the training takes about 42 hours in total simulation time and 2 hours in wall clock time for the policy to converge to achieve a robust performance.

## 6 Results and Comparisons

In this section, we first describe the performance of the gaits generated by the RL controller. Then, we compare our gaits to the scripted slithering gaits in terms of energy efficiency.

### 6.1 Results

In this study, 45 target velocities in the range of  $[0.025, 0.25]$  with a step interval of 0.005 are used for the evaluation of performance. Simulation results demonstrate that the PPO controller has succeeded in learning a gait from scratch without knowing any prior locomotion skills. Two exemplary gait patterns are shown in Figure 3. Surprisingly, we find that the PPO controller can adapt its gait pattern according to its desired speed like a real snake. The montages in the first row present a concertina-like gait at a low speed of  $0.05 \text{ m/s}$  and the second row resembles a slithering-like gait at a higher

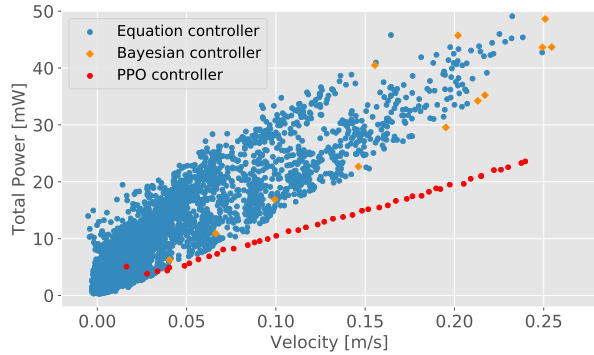


Figure 4: This scatter plot directly shows the energy consumption results of the controllers at a range of velocities. The blue point, orange diamond, and the red point stand for the data from the gait equation controller, Bayesian optimization controller, and the PPO controller, respectively.

speed of  $0.25 \text{ m/s}$ . In nature, snakes usually take the concertina locomotion at a low speed and switch to the slithering locomotion at a fast speed. In the second row, we also observe that the snake-like robot executes waves from the head to tail through a movement pattern of lateral undulation. This learned undulation wave is even smoother than the parameterized *serpentine* curve and drives the robot to move with better power efficiency.

The power consumption results of the learned gait are marked with red points in Figure 4. The data depicts a linear relationship between the travel velocity and the power consumption, which is in line with the physical law  $Power = Force \times Speed$ . There is only one point with higher power consumption when the velocity is around  $0.02 \text{ m/s}$ . Importantly, this also reveals the adaptability of the learning approach for generating gaits in a range of velocities. It can also be observed that the mean velocities do not exactly align with the specified interval of  $0.005$ , especially at higher target velocities. The reason for this is the difficulty to achieve the exact ratio between holding the right velocity and performing the corresponding power-efficient locomotion.

## 6.2 Comparison

We first have a look at the energy metric based the averaged power per velocity. After putting the power efficiency data of the *gait equation* controller (See Figure 2) and the PPO controller together, we can clearly conclude that the PPO controller has a much better power efficiency at a range of velocities (See Figure 4). As the velocity grows, the advantage of the PPO controller for saving energy is even more obvious. Taking the velocity of  $0.15 \text{ m/s}$  as an example, the PPO controller can save  $35\% \sim 65\%$  energy consumption depending on the parameter sets chosen by the *gait equation* controller.

We then discuss the power profile for each module of the robot. The power profiles of the slithering gait at the velocity of  $0.25 \text{ m/s}$  generated from the parameterized equation controller and the PPO controller are shown in Figure 5. It should be noted that we only present the results of the parameterized equation controller with the best energy efficiency at this speed. For the parameterized equation controller, the first joint consumes the most power since it rapidly adjust its head to aim at the front direction. Other module joints use similar power around  $5 \text{ mW}$  to  $5.7 \text{ mW}$ . For the PPO controller, the head joint still expends more energy than the average power

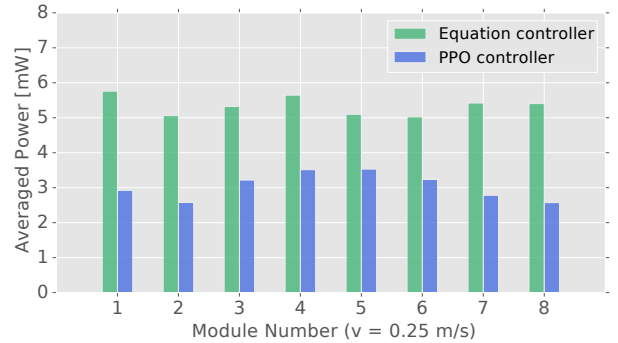


Figure 5: The power profile bar of the slithering gait at the velocity of  $0.25 \text{ m/s}$  for the equation controller and the PPO controller.

consumption. Besides, the joints close to the body center consume more power than the far-ends of the body. This is because the body trunk modules exert more strength to twist its body to generate the locomotion, which shows that the PPO controller generates more natural slithering gait compared to the parameterized slithering gait.

These superiorities can be elaborated from two aspects. On one hand, the traditional *gait equation* controller is based on the kinematics and describes the gait movement with no influence of physical forces such as friction or damping. Meanwhile, it only extracts several critical parameters to represent the gait, which is in fact a highly dimensional interaction with the environment. Although this parameterized description simplifies the control task, it inevitably brings difficulties for designing more sophisticated gaits despite of using optimization technologies, especially when the parameter space will grow exponentially with the increasing joint numbers. On the other hand, the RL method shows its effectiveness in the ability for solving this kind of complex control problem, since it is trained directly in the dynamic environment. It is able to generate an undulation gait that not only imitates real snakes, since natural evolution is not a perfect process but a compromise result. Therefore, it can explore its limitations and keep improving its behavior under its hardware constraints. This is a remarkable achievement because the algorithm has to handle a high degree of freedom without prior information about the environment or any locomotion behavior. In short, the PPO controller is able to overall outperform the equation controller in terms of power efficiency at a range of velocities.

## 7 Conclusion

Designing power-efficient gaits for snake-like robots remains a challenging task, since they come with redundant degrees of freedom and have complicated interactions with the environment. In this paper, we present a novel gait design method based on reinforcement learning. The learned gait has shown to have much better energy efficiencies at different travel velocities compared to the current kinematic-based method even after choosing those parameters using the grid search or Bayesian optimization algorithm. Our work contributes and serves as an exploration for designing sophisticated moving patterns for snake-like robots. Our future work will aim at designing gaits based on reinforcement learning for those kinds of snake-like robots without passive wheels.

## References

- [Calandra *et al.*, 2014] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian gait optimization for bipedal locomotion. In *International Conference on Learning and Intelligent Optimization*, pages 274–290. Springer, 2014.
- [Chernova and Veloso, 2004] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.*, volume 3, pages 2562–2567. IEEE, 2004.
- [Crespi and Ijspeert, 2008] Alessandro Crespi and Auke J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*, 24(1):75–87, Feb 2008.
- [Cully *et al.*, 2015] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015.
- [Dowling, 1996] Kevin J Dowling. Limbless locomotion: learning to crawl with a snake robot. *Unpublished Ph. D Thesis, The Robotics Institute, Carnegie Mellon University*, 5000, 1996.
- [Dowling, 1997] Kevin Dowling. *Power sources for small robots*. Carnegie Mellon University, 1997.
- [Gong *et al.*, 2016] Chaohui Gong, Daniel I Goldman, and Howie Choset. Simplifying gait design via shape basis optimization. In *Robotics: Science and Systems*, 2016.
- [Hirose, 1993] Shigeo Hirose. *Biologically inspired robots: snake-like locomotors and manipulators*, volume 1093. Oxford University Press Oxford, 1993.
- [Hu *et al.*, 2009] David L Hu, Jasmine Nirody, Terri Scott, and Michael J Shelley. The mechanics of slithering locomotion. *Proceedings of the National Academy of Sciences*, 106(25):10081–10085, 2009.
- [Kim and Uther, 2003] Min Sub Kim and William Uther. Automatic gait optimisation for quadruped robots. In *Australasian Conference on Robotics and Automation*, pages 1–3. Citeseer, 2003.
- [Kober *et al.*, 2013] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- [Kohl and Stone, 2004] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *AAAI*, volume 4, pages 611–616, 2004.
- [Kormushev *et al.*, 2018] Petar Kormushev, Barkan Ugurlu, Darwin Caldwell, and Nikos Tsagarakis. Learning to exploit passive compliance for energy-efficient gait generation on a compliant humanoid. *Autonomous Robots*, Feb 2018.
- [Liljebäck *et al.*, 2012] Pål Liljebäck, Kristin Ytterstad Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012.
- [Lizotte *et al.*, 2007] Daniel J Lizotte, Tao Wang, Michael H Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.
- [Long *et al.*, 2018] Pinxin Long, Tingxiang Fanl, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259, May 2018.
- [Ma, 1999] Shugen Ma. Analysis of snake movement forms for realization of snake-like robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 4, pages 3007–3013 vol.4, May 1999.
- [Melo and Paez, 2014] Kamilo Melo and Laura Paez. Experimental determination of control parameter intervals for repeatable gaits in modular snake robots. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pages 1–7, Oct 2014.
- [Peng *et al.*, 2017] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics*, 36(4):41, 2017.
- [Rajeswaran *et al.*, 2017] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint 1709.10087*, 2017.
- [Saito *et al.*, 2002] Masashi Saito, Masakazu Fukaya, and Tetsuya Iwasaki. Modeling, analysis, and synthesis of serpentine locomotion with a multilink robotic snake. *IEEE control systems magazine*, 22(1):64–81, 2002.
- [Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.
- [Tesch *et al.*, 2009] Matthew Tesch, Kevin Lipkin, Isaac Brown, Ross Hatton, Aaron Peck, Justine Rembisz, and Howie Choset. Parameterized and scripted gaits for modular snake robots. *Advanced Robotics*, 23(9):1131–1158, 2009.
- [Tesch *et al.*, 2011] Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1074, Sep. 2011.
- [Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.
- [Tucker, 1975] Vance A. Tucker. The energetic cost of moving about: Walking and running are extremely inefficient forms of locomotion. much greater efficiency is achieved by birds, fish—and bicyclists. *American Scientist*, 63(4):413–419, 1975.
- [Yu *et al.*, 2018] Wenhao Yu, Greg Turk, and C Karen Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37(4):144, 2018.