# Bioinspired Dynamic Control of Amphibious Articulated Creatures with Spiking Neural Networks

Ioannis Polykretis*
Accenture Labs
Rutgers University

Mridul Aanjaneya*
Rutgers University

Konstantinos Michmizos†
Rutgers University

**ABSTRACT**

We present a biologically plausible, compact spiking neural network for controlling the crawling and swimming behaviors of amphibious creatures with articulated skeletons. Prior methods for learning efficient control policies for such creatures are resource-greedy, both in terms of computational time and energy requirements due to the high number of degrees of freedom introduced by the many joints present in the creature skeleton. Our approach takes a radical departure from prior work and exploits the physiology of amphibious creatures. Specifically, we emulate experimentally identified biological controllers for amphibious creatures with a network of spiking neurons, which alleviates the need for training altogether and can potentially provide the additional benefit of utilizing minimal resources in terms of energy. Our approach is robust and allows the amphibious creature to avoid both static and dynamic obstacles when exhibiting different movement patterns, and also adaptively control its swimming speed. Moreover, we show that the creature can seamlessly transition between crawling and swimming behaviors as it moves from land to water or vice-versa, similar to its real-world counterpart. Our approach presents an efficient and scalable alternative for producing vivid and lively motion, as we demonstrate through a complex scene where multiple amphibious creatures interact with each other, successfully avoiding collisions while moving across a pool of water. Our approach is generalizable to other creatures also, as we show through the design of a controller for a quadruped.

**Index Terms:** Computing methodologies—Computer graphics—Physical Simulation—

## 1 INTRODUCTION

Character animation is an integral part of computer graphics [27, 62, 68]. Beyond the immense success in the VFX industry [31, 32], methods for controlling characters are finding increasing use cases in robotics for applications such as housekeeping [23], autonomous driving [69], meal assistance [47], etc. This development is partly motivated by the promise that robots may some day completely replace humans in dangerous or undesirable tasks, such as rescue missions [48], and planetary [54] or underwater explorations [52]. While wheeled robots can effectively address simple tasks in structured environments, the increasing variability in unstructured environments demands articulated robots with multiple degrees of freedom (DOFs). However, this additional complexity also makes it more challenging to control their behaviors.

The most common and successful method for training the behaviors of articulated characters is Reinforcement Learning (RL) [4, 28, 49]. RL algorithms have provided remarkable results in a wide range of tasks spanning from high-level cognition, such as path planning [72], to low-level control of crawling [6], walking [39], and swimming [46, 57] behaviors. However, RL methods also suffer from some crucial drawbacks. First, their training phase requires tens of thousands of steps, even for simple tasks such as an inverted pendulum [20], resulting in considerably long training times. Second, the training time also scales with the number of DOFs of the characters [64], rendering the training of such agents computationally challenging. Third, the prolonged training results in significant power consumption [17, 41].

Decreasing the training time of articulated characters/robots, both algorithmically and computationally, and lowering power requirements has been a major goal for the research community, and has led to the development of step-skipping algorithms [35] and specialized hardware accelerators [59].

Alternatively, researchers have drawn inspiration from biological agents that effortlessly give rise to complicated behaviors and have attempted to replicate them in bioinspired algorithms for character control [9, 38, 49]. Especially for repetitive behaviors, such as locomotion and swimming, numerous studies have replicated the function of the specialized biological networks called *Central Pattern Generators* (CPG) [30, 57] that are known to drive such behaviors. These networks give rise to periodic activity patterns that drive the limbs and muscles of biological agents for generating the motion [25]. The periodic patterns, which are the core of such networks, have been traditionally approximated with oscillators defined as dynamical systems, alleviating the need for training altogether [29, 30]. However, arbitrary sets of ordinary differential equations (ODEs) that simulate oscillators require precise numerical operations. As a result, they are not inherently compatible with edge computing hardware, such as neuromorphic processors [14, 43], that could also alleviate the power requirements of the control. Even when explicitly designed for deployment on such platforms to benefit from their energy efficiency, ODE-based methods introduce additional resource requirements [22, 60].

In this paper, we exploit the identified biological principles of neuronal CPG networks that give rise to crawling and swimming behaviors in amphibious animals, such as lizards, salamanders, and crocodiles. We design a compact network of spiking neurons, which emulates the individual biological neurons and their connectivity in CPG networks to control the motions of an articulated amphibious creature without the need for any training. Our control method produces biologically realistic motion for one crawling and two swimming patterns, including the ability to seamlessly transition between these behaviors. Additionally, we demonstrate that the amphibious creature can adapt its speed to external control signals. We also incorporate sensory feedback to allow the creature to avoid collisions with static and dynamic obstacles that may be present in the surrounding environment. In summary, our main contributions are as follows:

- A continuous-valued oscillator that utilizes the aggregate population dynamics of groups of bursting neurons,
- Three novel, bioinspired architectures based on spiking neural networks (SNN) that use populations of bursting neurons as building blocks for controlling two different swimming patterns and one crawling pattern of an articulated amphibious creature, and seamlessly transitioning between them, and

*e-mail: {ip211, mridul.aanjaneya}@rutgers.edu
†e-mail: km1078@cs.rutgers.edu

- A closed-loop controller that responds to external stimuli and modifies the creature's speed and direction of motion to navigate in unstructured environments with static and dynamic obstacles.

Our bioinspired control method requires the understanding of the biological networks that drive the swimming behaviors in animals and some empirical tuning of its hyperparameters in exchange for a prolonged training that is conventionally necessary for driving the amphibious creature. Our work hints that biologically plausible control of low-level behaviors with networks of spiking neurons could mitigate the computational burden of training multi-DOF agents with RL and allow for parallel employment of the available computational resources for performing high-level cognitive tasks.

## 2 Related Work

Simulating biologically plausible motion of articulated characters is an active area of research in computer graphics [11, 12, 18]. Its primary goal is the demonstration of physically realistic behaviors, which can preserve the robustness of their biological counterparts to perturbation and which can adapt to control inputs. While biped locomotion has been extensively studied for understanding the emergence of human gait patterns [37, 38], and also their recovery [56] as well as enrichment with fine motor skills [42], the walking motions of multi-legged animals has also attracted considerable interest [19, 44, 51]. Researchers have also explored swimming of humanoids [57] and aquatic animals [46], and flying in bird-like characters [34, 67]. The common ground in all these works is the exploration of the underlying control algorithms that can give rise to these fine and complex movement patterns.

Increasing the number of DOFs in characters and the complexity of the surrounding environment results in high-dimensional observation and action spaces, which cannot be easily captured in predefined datasets of reasonable size. This highlights the applicability of RL algorithms that are based on trial-and-error for accumulating rewards and punishments [4, 28]. The inclusion of deep network architectures has provided remarkable generalization to diverse agents and tasks, such as the generalization of policies from biped humanoids to dinosaur-resembling quadrupeds [49], or the control of diverse soft-bodied animals spanning from starfishes to octopuses and from stingrays to squids [46]. Beyond the generalization to different agents, RL algorithms can also address various tasks. For example, the deep networks in [49] are not limited to walking control and can also replicate dancing, flipping, and even throwing behaviors. However, this versatility comes at a significant computational cost. Even for small and simple tasks such as an inverted pendulum, the training requires tens of thousands of steps [20]. The training time deteriorates with increase in the number of DOFs [64], as hundreds of thousands of samples are required to achieve the required generalization. Moreover, long training processes need to be repeated even when only slight changes are made to the task or the character's skeleton [21]. In addition to compute time, long training sessions are considerably greedy in terms of power consumption [17, 41], which is an essential concern for mobile and resource-constrained agents [63].

To mitigate the computational burden of RL training, a line of research sought inspiration in the effortless emergence of repetitive behaviors such as locomotion, swimming, and flying in biological agents. Typically, such methods approximate the biological CPG networks with oscillator circuits [30, 71]. The resulting controllers require minimal training for effectively controlling the character's joints and give rise to several motion patterns. The output of such networks is a sinusoid that could, theoretically, be replaced by a simple function generator. However, a static sinusoidal function is insufficient to provide effective control with real-time adaptation (frequency, phase, continuity) to dynamic environments. Conversely, the dynamical systems can adapt to external signals [30] and can also incorporate learning to fine-tune their behavior [57]. However, they also introduce some new limitations to the control problem. First, the generation of the oscillatory behavior relies on systems of high-order, non-linear differential equations [57, 66], whose solution requires high-order numerical methods and high precision, which introduces computational overhead. More importantly, the use of ODE systems limits their applicability to accelerated and energy-efficient edge devices, such as neuromorphic processors. This incompatibility is due to the biological inspiration of these methods being limited to the phenomenological level of the oscillatory activity. A handful of works [2, 40, 50] attempt to go deeper and emulate the characterized underlying neuronal connectome of the CPG networks that drives the periodic activity [25]. These methods benefit from the compact and explainable connectivity of the biological networks to provide effective and efficient control, but this area is still in its infancy.

Our work follows this last direction and alleviates the need for training when controlling the crawling and swimming motions of articulated amphibious creatures. Emulating the experimentally identified connectome of the CPG that drives these behaviors in animals such as salamanders and lizards, our compact SNN produces different behaviors and naturally transitions between them. Using a small number of spiking neurons and a simplified model for each of them, we reduce the computational requirements and provide a controller that can be directly deployed on different neuromorphic platforms, allowing for energy-efficient character control.

## 3 Bursting Neuron Model

Neurons in artificial neural networks sum continuous-valued inputs, apply threshold functions and provide continuous-valued outputs [7]. Therefore, they only loosely approximate biological neurons, which integrate their synaptic inputs in their continuous membrane voltage. When this voltage exceeds a threshold value, the neuron emits a binary, all-or-none output called a *spike* and resets its membrane voltage. Adding more functionality to their networks, different types of spiking neurons have distinct firing modes, with a prominent one being the *bursting neuron* [36].

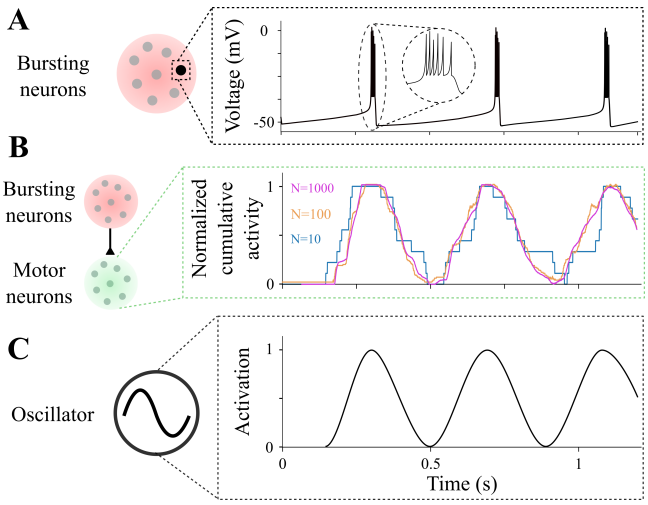


Figure 1: The activity of a single bursting neuron alternates between spike bursts and quiescence and can approximate discrete signals (A). The cumulative activity of a population of sufficiently many bursting neurons (B) can closely approximate a continuous sinusoidal oscillation (C) as the number of neurons increases.

Unlike regular-firing neurons that emit solitary spikes and reset, bursting neurons fire a packet of spikes called a *burst* when exceeding their threshold and reset for a more extended period of quiescence (see Figure 1A). This alternation between spiking and quiescence, which is not typical in neurons with other firing modes,

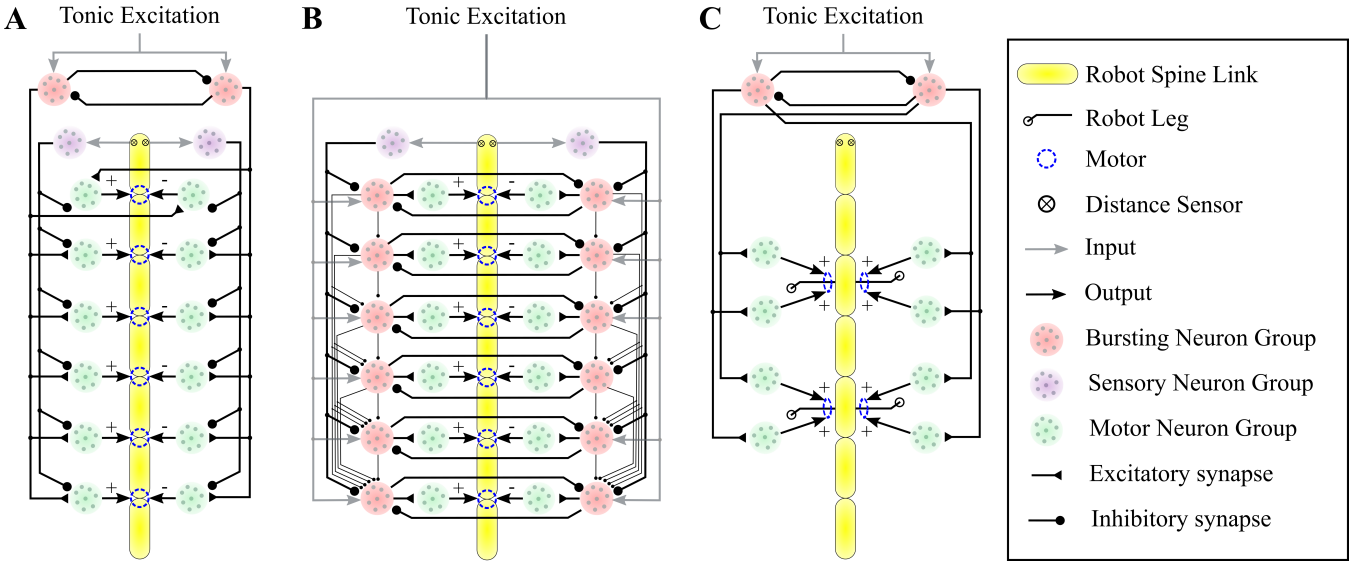


Figure 2: **A.** A simple controller driving alternating contractions of the left and right sides of the creature's body. The contractions of the first joint are anti-phasic with the rest of them to drive the forward swimming movement. **B.** A controller consisting of a chain of oscillators propagates the activation sequentially through the creature's body and gives rise to an undulatory movement. **C.** A controller that rotates the creature's legs forward. When combined with the alternating spine contractions (A), the movement of the legs generates the crawling pattern.

makes the bursting neuron a great candidate for approximating the periodic fluctuation of an oscillator. In fact, a single bursting neuron has been used to drive the discrete, periodic movements (such as, elevation and lowering, forward and backward motion) of a hexapod's limbs and control its locomotion [50].

However, the activity of a single bursting neuron resembles a binary oscillator that toggles between two discrete states. While this binary nature is sufficient for driving motion patterns that consist of discrete components (such as, leg elevation/lowering), it results in abrupt movements and discontinuous behaviors. Consequently, it hinders the applicability of the bursting neuron for continuous control tasks that require intermediate position values over a range of the discrete extremes. To address this issue, we propose a continuous-valued oscillator that utilizes the population dynamics of groups of bursting neurons, as shown in Figure 1B.

## 4 Population of Bursting Neurons

Before describing our approach for simulating the spiking activity of a population of bursting neurons, we explain our modeling of a single bursting neuron. We used the simplified neuron model in [33] which is governed by the following set of equations:

$$\dot{v} = 0.04v^2 + 5v + 140 - u - I_{ex} + \frac{1}{M}\sum_{j=1}^{M} s_j w_j v_{psp} + \varepsilon, \quad (1)$$

$$\dot{u} = a(bv - u), \quad (2)$$

with the voltage being reset based on the following condition:

$$\text{if } v \geq 30 \text{ mV, then } v \leftarrow c,\ u \leftarrow u + d, \quad (3)$$

where $v$ is the neuron's voltage, $u$ is an internal state variable, $\dot{v}, \dot{u}$ denote time derivatives, $I_{ex}$ is an external input current, $s_j \in \{0,1\}$ denotes the spike of the $j$th presynaptic neuron, and $w_j$ is the weight of the connection for the $j$th out of the $M$ presynaptic neurons. The postsynaptic potential $v_{psp}$ (set to 1.5 mV) is the increase in the postsynaptic voltage membrane due to a single presynaptic spike, while $\varepsilon \sim U(-0.2, 0.4)$ is a uniformly distributed noise factor used for emulating the variability of the population dynamics that emerge from weight mismatch, synaptic failures, etc [1]. The parameters $a$, $b$, $c$, and $d$ define the neuron's firing mode. We set $a$ to 0.02, $b$ to 0.2, $c$ to -65, and $d$ to 2 as suggested in [33] for simulating all neurons. For simulating the slow bursting behavior (see Figure 1A), we set $a$ to 0.00125 to slow down the decay of $u$, and $c$ to $-47$ to reset $v$ at a higher value. We integrate equations (1) and (2) using forward Euler with a time step of 0.5 ms.

Next, we aggregate the activity of multiple bursting neurons and observe their population dynamics, as shown by the motor neurons in Figure 1B. Although small populations of neurons (e.g., $N = 10$) introduce intermediate levels of oscillatory activity, increasing the population size (e.g., $N \in \{100, 1000\}$) improves the approximation of a continuous-valued oscillator, as shown in Figure 1C. Even in its most resource-greedy version (1000 neurons per population for 12 bursting neurons, 12 motor neurons, and 2 sensing populations, 26K in total), our controller, can still be deployed on a single neuromorphic processor (Intel's Loihi includes 128 neurocores of 1024 neurons each [14]). In fact, that size of our SNN is comparable to similar controller networks proposed before (38 neurocores in [60]), whose power consumption is about 10mW. In a real-world system, the power required for the motors is generally dominating the overall system consumption. However, our method targets the computational aspects of control; hence, a fair comparison can be made against other methods with the same goal, i.e., planning the motor activations, integration of sensory feedback and behavior adaptation, etc. In that domain, the use of spiking neurons with minimal training requirements does provide considerable improvement in energy consumption. As a result, the biological plausibility of our network and its compatibility with edge-computing hardware provide significant energy efficiency for the control task in a dynamic environment.

## 5 Bioinspired Motion Control

Using a population of bursting neurons as a building block, we propose three novel, bioinspired architectures based on spiking neural networks (SNN) to control an amphibious articulated creature: (a) the sequential alternation network (Figure 2A) that gives rise to alternating, synchronized contractions of the right and left sides of the creature's body, (b) the chained oscillator network (Figure 2B) that gives rise to an undulatory movement of the spine by propagating the activation of the joints along the spine, and (c) a subnetwork that we use in conjunction with the sequential alternation controller to move the creature's legs (Figure 2C) during crawling.

### 5.1 Sequential Alternation Controller for Swimming

This controller draws inspiration from the alternating flaps of the tails of a fish when swimming [24]. Two mutually inhibiting bursting neuron populations acting as coupled oscillators form the core of the sequential alternation controller, as shown in Figure 2A. We drive these two populations with an external input that dictates their oscillation frequency. This input corresponds to control signals from the midbrain [10] that descends to the spine in the form of tonic excitation, a stable excitatory input [58]. To ensure that the spiking activity of the two populations will be alternating and not concurrent, we fully connect them with inhibitory synapses whose weights $w$ are set to -1. In this way, the population that is activated first delays the activation of the other, and so on.

We use two groups of motor neurons for each spinal joint, corresponding to the flexor and extensor motor neurons, for translating the alternating activation of the two bursting populations to alternating contraction of the right and left sides of the creature's body. For driving the alternating contractions, we use ipsilateral excitatory connections with a weight $w$ set to 1 from one bursting population to the flexor motor neuron, and from the other to the extensor motor neuron. For ensuring a forward swimming movement, we mimic the experimentally identified connectivity of CPG in fish [25, 55] that moves the creature's head in a direction opposite to that of the rest of its body. This is achieved by introducing contralateral connections from the bursting neuron populations to the motor neurons only for that joint. The spikes of the flexor motor neuron increases the joint's angle, leading to a left contraction, while the spikes of the extensor motor neuron decreases it, leading to a right contraction. For controlling the joint angle using the motor neuron spikes, we first calculate the normalized, instantaneous motor neuron activity as follows:

$$\alpha = \frac{1}{N}\left(\sum_{j=1}^{N} s_j^F - \sum_{j=1}^{N} s_j^E\right), \tag{4}$$

where $s_j^F$ and $s_j^E$ denote the spikes of the flexor and extensor motor neuron populations respectively, and $N$ is the population size. Then, we use $\alpha$ to modify the joint's angle $\theta$ as follows:

$$\theta \leftarrow \theta + \alpha, \theta \leftarrow \langle -\theta_{max}, \theta_{max} \rangle, \tag{5}$$

where $\langle\cdot\rangle$ denotes the clipping of the angle value between its minimum and maximum value. We set $\theta_{max}$ to $\pi/8$ to generate biologically realistic and aesthetically pleasing motion sequences.

### 5.2 Chained Oscillator Controller for Swimming

This controller is inspired by the undulatory movement of the spine in eels, lizards, and snakes [55], which is achieved by contracting (i.e., by flexing or extending) all the spinal joints in the same direction. However, unlike the sequential alternation controller, these contractions are not synchronized but have a phase lag.

To control this movement, we introduce one pair of bursting neuron populations per joint (see Figure 2B). These populations act as coupled oscillators for controlling each joint independently from the others. We drive all bursting neuron populations with the same tonic excitation input that dictates their oscillation frequency.

For simulating the required phase lag in the contractions of the spinal joints, we break the synchrony of the different oscillators by introducing top-down inhibitory connections from the oscillators closer to the creature's head to the ones closer to its tail, which are illustrated by the thin lines in Figure 2B. The weights $w$ of these connections are set to $-0.5/N_{tdi}$, where $N_{tdi}$ is the number of top-down inhibitory inputs that each bursting neuron population receives. In this way, the activation of the first oscillator inhibits the rest, introducing a slight delay to the onset of their activity until their input exceeds the inhibition. With each bursting neuron population inhibiting the ones below it, they are activated one after the other and the oscillations exhibit the desired phase lag.

For translating these orchestrated oscillations of the bursting neuron populations to joint contractions, we introduce flexor and extensor motor neuron populations, as described in Section 5.1. We use one bursting neuron population of each mutually inhibiting pair to drive a flexor motor neuron population which increases the joint's angle, and the other to drive an extensor motor neuron population which decreases it. For enforcing the contractions of all joints in the same direction, we only introduce ipsilateral connections between the bursing neuron and the motor neuron populations.

### 5.3 Leg Controller for Crawling

For moving the creature forward when it is on the ground, we utilize the rotational motors for its legs in conjunction with the sequentially alternating spine movement (as described in Section 5.1) during crawling. To drive the forward crawling, we group the four legs into two pairs. The left front and right hind legs form one pair, and the right front and left hind legs form the other pair. The legs in each pair are synchronously rotated forward and we introduce a constant phase lag of a half-cycle to enforce the alternating movement of the two pairs. To control each leg, we employed two motor neuron populations similar to the control of the spinal joints, as shown in Figure 2C. To ensure the synchronization of each leg pair with the spine contraction in one direction, we connected the two bursting neuron populations of the sequential alternating spine controller to the two motor neuron populations that control the legs. We set the weights of these excitatory connections to 1. To achieve forward crawling, we rotate each leg only in the forward direction by translating the spikes of both motor neuron populations to an increase in the joint angle. We set $\theta_{max}$ to $\pi$ for the leg joints. Note that the angle of the leg joints was not clipped using equation (5) to allow for continuous forward rotation and crawling movement.

### 5.4 Transition between Motion Patterns

For transitioning between the crawling and swimming patterns, or vice-versa, we use the creature's position in the surrounding environment as provided by a GPS sensor. When it is no longer on the ground and sufficiently submerged into the water based on a user-specified tolerance, we stop the movement of its legs and force them to point backwards for minimal resistance while swimming. Similarly, we use the leg controller when the GPS sensor indicates that the creature has moved sufficiently out of the water.

## 6 Sensors and Feedback Networks

With a few exceptions [3, 61], SNN-based CPG are open-loop controllers that do not modify their behavior in response to sensory inputs [16, 26]. While some of them can adapt their oscillation frequency to external signals [50], they still disregard sensory feedback, hindering their applicability to dynamic environments where the agent needs adapt its behavior.

To remedy this, we introduce two feedback loops to our SNN-based CPG to allow the amphibious creature to interact with its surrounding environment. The first feedback loop utilizes a GPS sensor, from which we read the creature's position and swimming speed in real-time. This allows us to select the used motion pattern as a function of its position. The second feedback loop employs two distance sensors that are positioned on the sides of the creature's head, resembling a pair of eyes (see Figure 2) and allowing it to detect and avoid obstacles in its environment. Each distance sensor is equipped with six rays arranged equidistantly on the surface of a cone, whose angle is 45 degrees, with a maximum sensing distance $d_{\max} = 0.7$ m, and provides a value $d$ that is an estimate of the distance between the creature's head and the closest object.

### 6.1 Distance Sensory Feedback Network

To incorporate the distance sensory feedback, we translate the readout values $d_r$ and $d_\ell$ of the right and left sensor, respectively, into two excitatory current values as follows:

$$I^r_{ds} = A \cdot \big((d_{\max} - d_r) - (d_{\max} - d_l)\big), \quad (6)$$

$$I^l_{ds} = A \cdot \big((d_{\max} - d_l) - (d_{\max} - d_r)\big), \quad (7)$$

where $A$ is a scaling factor that is set to 70 to rescale the current value to the operating range of the neuron model. Next, we use these excitatory currents to drive two sensory neuron populations (see Figures 2A and B), whose spiking activity encodes the distance of objects from the left and right sides of the creature's head.

Subsequently, we use the spiking activity of these two distance sensory neuron populations to modify the output of the CPG network. In the sequential alternation controller, we decrease the contraction of the spinal joints in the direction of an obstacle by connecting the sensory neuron to the flexor and the extensor motor neuron populations of the spinal joints with inhibitory synapses, whose weight $w$ are set to $-1$. In this way, when an obstacle is on the left (right) side of the creature's head, the left (right) distance sensor provides a readout value $d_\ell < d_{\max}$ ($d_r < d_{\max}$). This increases the current value $I^r_{ds}$ ($I^\ell_{ds}$) that stimulates the corresponding sensory neuron populations (see equations (6) and (7)) and increases the sensory neuron activity. The sensory neuron spikes inhibit the motor neuron populations that induce left (right) contractions, turning the creature to the right (left) and avoiding the obstacle. We achieve a similar result with the chained oscillator controller by connecting the sensory neuron populations directly to the bursting neuron populations to inhibit their activity and decrease the contractions in the given direction, as described above. Note that in the sequential alternating controller, we connect the sensory neuron populations directly to the motor neuron populations, not the busting neuron populations. This was done so as to avoid interference of the sensory feedback network with the leg movement network since the bursting neuron activity in that controller directly reflects on the frequency and range of the leg movements.

## 7 Generalization to Quadruped Locomotion

To demonstrate the applicability of our approach to agents beyond the aforementioned articulated creature, we sought to use our method to control the locomotion of a quadruped. As an example, we focused on the *Ant-v2* agent from *MuJoCo OpenAI gym* that is widely used as a test bed for RL algorithms. Similarly to the salamander controller, a pair of mutually inhibiting bursting populations constituted the core of the quadruped controller and its specific connectivity was then dictated by the desired locomotion pattern. More specifically, the legs of the quadruped were grouped in two pairs (front left with hind right and front right with hind left), with sequentially alternating half-cycles: The first half-cycle consisted of the lifting and forward motion of the first pair, and the concurrent lowering and backward motion of the second pair, while in the second half cycle the order of the pairs switched. To recreate this behavior, we utilized one pair of mutually inhibiting BN populations, whose activity set the pace of the sequential alternation (Fig. 3). Each BN population (e.g., left) drove a pair of MN groups (e.g., top-left and bottom-right). Each MN group lifted and moved forward the corresponding leg, while lowering and moving backward a leg of the opposite pair. To compare the performance of our SNN controller, we used a RL baseline and trained the quadruped to walk using the well-established Proximal Policy Optimization (PPO) algorithm [15, 53]. We measured the training time (number of episodes) it required to reach the SNN performance, and compared the reward accumulated by the SNN controller and PPO in the early episodes of the training.

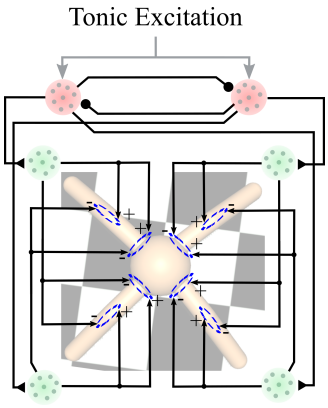


Figure 3: Architecture of the quadruped locomotion controller.

## 8 Simulation Results

We used the Webots framework [45] for our simulations, an open-source mobile robot simulation software developed by Cyberbotics Ltd. We chose this platform because it has been successfully used by other research groups in prior work for simulating swimming and amphibious robots [8, 13, 70]. We used the salamander robot prototype [30] and environment that are provided with this software package. Upon acceptance, accompanying this article, we will open-source our code for running 3D simulations of amphibious creatures with our proposed bioinspired controller.

In this section, we demonstrate the applicability of our proposed method for controlling the behavior of an amphibious articulated creature. Our method can effectively generate two swimming patterns, as described in Sections 8.2 and 8.4, and one crawling pattern (see Section 8.3) for the amphibious creature. Moreover, our controller is robust enough to adapt the creature's speed to external control signals, as described in Section 8.6, and drive it to avoid collisions with static and dynamic obstacles in its surrounding environment (see Section 8.7). We first present how our spiking neural network can control the contractions of a single spinal joint.

### 8.1 Single Joint Control

We control the alternating contractions of a single spinal joint in a simplified version of the amphibious creature with the core module of our spiking neural network, as shown in Figure 4. The external tonic excitation input induces bursting activity in both bursting neuron populations, and their mutual inhibition ensures the alternation of their bursting activities. With each bursting neuron population exciting a corresponding motor neuron population, the activity of the two motor neuron populations is also alternating (see Figure 5A). Translating the motor neuron activity to changes in the joint's angle (equations (4)–(5)) results in an oscillation between the minimum and maximum angle values, as shown in Figure 5C. This oscillation of the joint angle translates to an alternating contraction of the right and left sides of the creature's body, as illustrated in Figure 5B. As these alternating contractions of the two sides of the body constitute the basis of swimming movements, we move on to examine the emergence of such swimming patterns with our full-scale controllers.

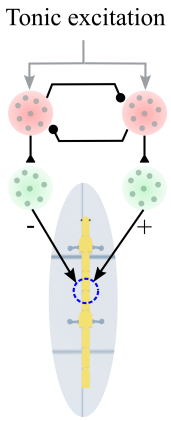


Figure 4: SNN for a single joint.

### 8.2 Sequential Alternation Swimming Pattern

We extended the core module of our spiking neural network to control all six spinal joints of the amphibious creature for generating a swimming pattern with alternating contractions of the right and left sides of its body, as shown in Figure 6. Our controller (see Figure 2A) drives each of the six joints as described in Section 8.1, giving rise to similar oscillations of their angles between the minimum and maximum values. The oscillations of different joints are synchronized and give rise to synchronous contractions of the left side of the body that alternate with synchronous contractions of the

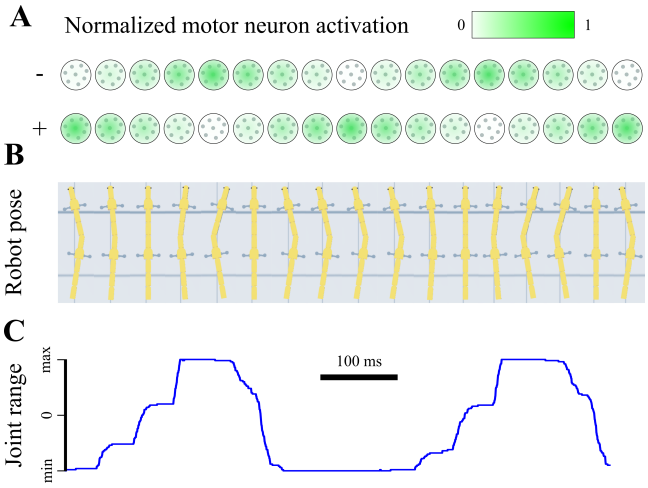


Figure 5: Alternating contraction of a single joint in a simplified creature model using our SNN oscillator. Two alternating bursting neuron populations drive the activity of two motor neuron populations that flex and extend the joint. The positive motor neuron population (right in Figure 4) increases the joint angle (C), while the negative motor neuron population (left in Figure 4) decreases it. The joint flexion and extension changes the creature's pose (B).

right side (see the bottom row in Figure 6). This synchronization emerges from using a single pair of mutually inhibiting bursting neuron populations that act as a coupled oscillator and dictate the pace of the contractions for the whole spine. While synchronized with the rest of the spinal joints, the contractions of the first joint are anti-phasic (see top row in Figure 6). This constant phase lag of 180 degrees is due to the contralateral connections of the oscillatory bursting neuron populations to the flexor and extensor motor neuron populations of the first joint, as shown in Figure 2A. Functionally, this phase reversal moves the creature's head in the opposite direction to the rest of its body and creates the necessary propulsion to move in the water. Note that the short, synchronized joint activation in the first few milliseconds of the movement (see Figure 6) is an artifact due to the initialization of the neuron stimulation.

### 8.3 Crawling Pattern

To allow the creature to move on the ground, we combine the alternating contractions of its spine with the rotation of its legs via a subnetwork that actuates the legs, as shown in Figure 2C. The movement of the legs is pairwise synchronized: the right front leg and the left hind leg rotate simultaneously and, consequently, always point in the same direction (see Figure 7A). The other two legs are similarly synchronized. The synchronization of the leg rotations with the sequential alternation of the spine contractions gives rise to the crawling pattern. More specifically, the two pairs of legs complete a half-rotation with each spine contraction. In this way, the right front and left hind legs always point forwards when the spine contracts to the left, as shown in Figure 7A1, while the other two legs point backwards, and vice-versa. This synchronization of the leg rotations and the spine contractions is due to the common drive of their corresponding motor neuron populations by the same oscillatory core, i.e., the pair of mutually inhibiting bursting neuron populations, as shown in Figures 2A and C. The crawling motion allows the creature to initially move on the ground before entering the water (see Figure 7B). Its legs keep rotating as long as the water is shallow and then stop and are kept to the side of the body when only a swimming movement is required. This transition from the crawling to the swimming motion is dictated by the estimate of the creature's position, as provided by the GPS sensor.

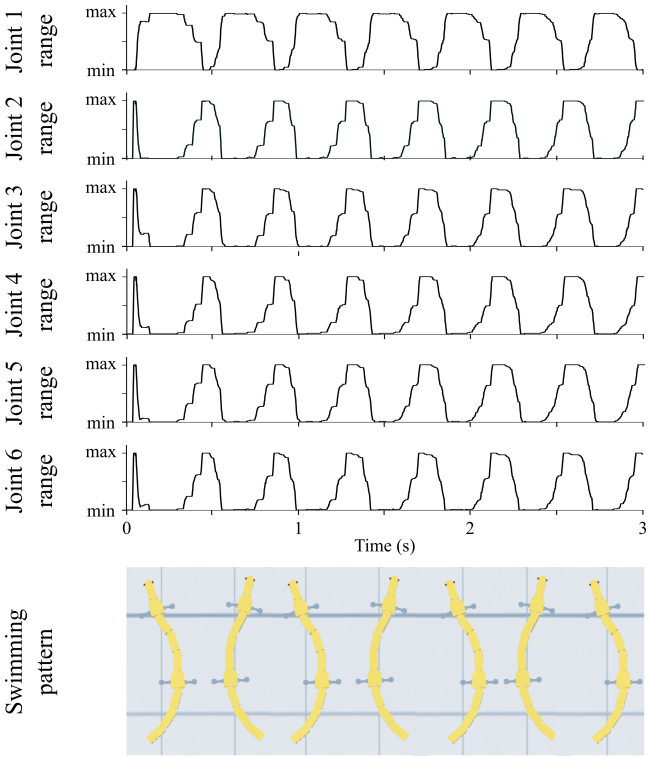


Figure 6: Swimming pattern with alternating contractions of the left and right sides of the creature's body. The sequential alternation controller presented in Figure 2A drives the six rotational joints on the spine. The phase reversal of the first joint (top) generates the required propulsion for the swimming motion (bottom).

### 8.4 Undulatory Swimming Pattern

In this section, we demonstrate how our chained oscillator controller, as shown in Figure 2B, gives rise to the characteristic undulatory movement of various amphibious animals, such as salamanders, crocodiles, and lizards. Similar to the swimming pattern with sequentially alternating contractions (see Figure 6), our spiking neural network induces oscillations for all six spinal joint angles between their maximum and minimum values, as shown in Figure 8. However, these oscillations are not synchronized. Instead, there is a phase lag between the oscillations of consecutive joints (shown by the dashed line in Figure 8). This phase lag is due to the top-down inhibition, as described in Section 8.4. Functionally, this results in the propagation of contraction initiation along the spine and gives rise to the undulatory movement, as shown in the bottom row of Figure 8. Notably, the magnitude of this phase lag adapts to the oscillation frequency, allowing for undulatory swimming at different velocities without any changes to the network architecture or configuration. In agreement with experimental findings [5, 65], we diminish the phase lag at low and high oscillation frequencies, breaking the swimming pattern (see also Section 9). We also highlight that the orchestrated joint contractions generate the propulsion to move the creature, without requiring an anti-phasic joint oscillation as used in the sequential alternating controller (see Section 8.2). The short, synchronized joint activation in the first few milliseconds of the movement (see Figure 8) is an artifact due to the initialization of the neuron stimulation.

### 8.5 Timings

We examined the applicability of our method to real-time character control by measuring the execution time of our SNN controller. A stripped-down implementation of our sequential alternation network

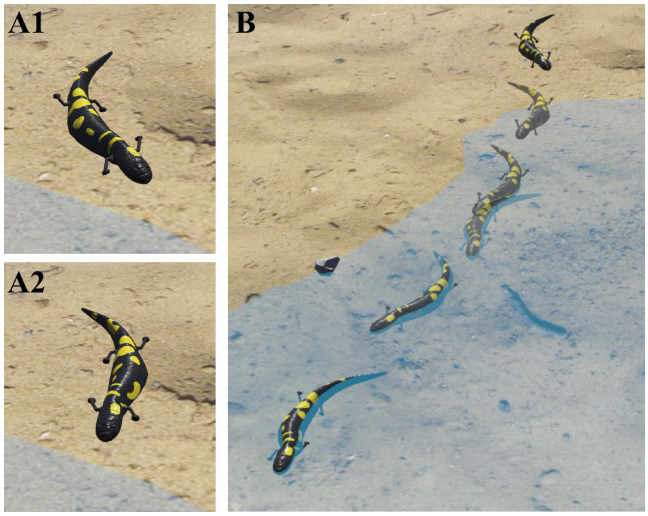


Figure 7: The crawling pattern of the amphibious creature (A) moves it on the ground until it reaches the pond, where it transitions to the swimming pattern (B). During crawling, the contractions of the left side of the body (A1) are synchronized with the front right and hind left legs pointing forward, while the two remaining feet are synchronized with the contractions of the right side (A2).

(see Section 5.1) simulated 1s of real-time in $689,63 \pm 0.38$ ms, while the larger chained oscillator network (see Section 5.2) required $829,07 \pm 0.72$ ms. These results are averaged over 100 iterations running on an Intel i5-1035G1 CPU with 8 cores at 1 GHz.

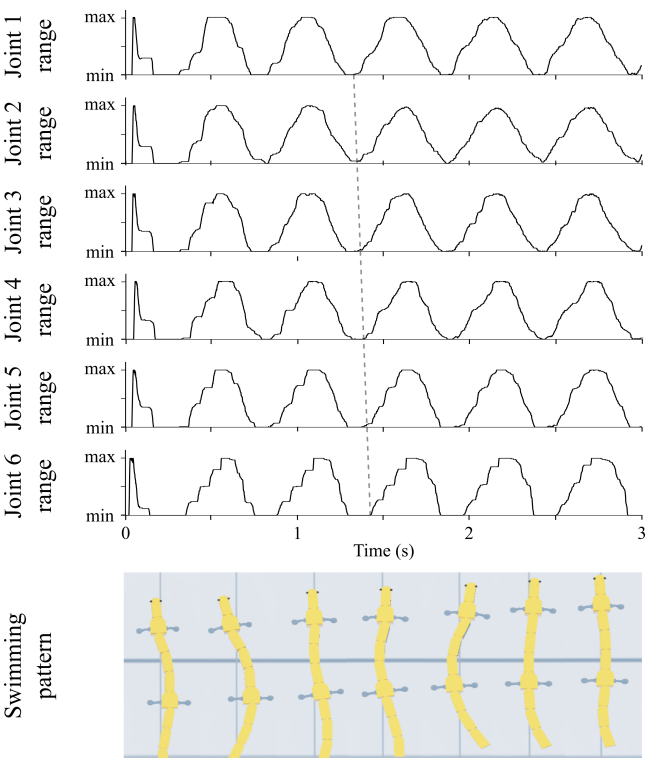


Figure 8: The chained oscillator controller (see Figure 2B) drives the six rotational joints of the spine. The phase lag (dashed line) introduced by the top-down inhibition gives rise to the characteristic undulatory swimming motion of amphibious creatures (bottom).

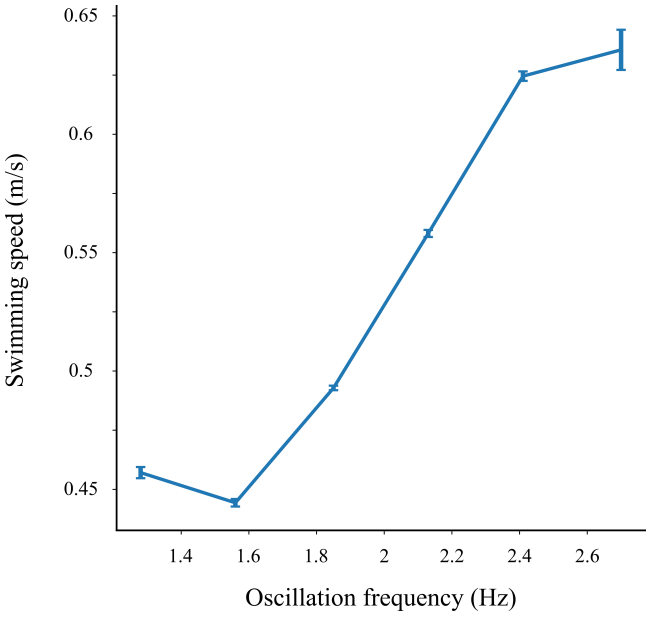


Figure 9: Adaptation of the creature's swimming speed to external input. The frequency of the network's oscillation adapted to the tonic excitation input. In turn, the oscillation frequency controls the swimming speed. Notably, the swimming speed saturates at both low and high oscillation frequencies, while providing a linear control range as observed in real animals [55]. Error bars denote standard deviation of the mean swimming speed over 10 iterations.

### 8.6 Speed Adaptation

The periodic nature of the oscillations raises the question whether its possible to control their frequency and, thus, the creature's swimming speed. To investigate this, we focus on the undulatory swimming pattern, which is utilized by animals when fast swimming is required, as it can produce high thrust forces [55]. To modify the oscillation frequency, we provide different levels of tonic excitation to the mutually inhibiting bursting neuron populations shown in Figure 2. More specifically, we select six levels of input current $I_{ex}$, which cover the operating range of the simplified neuron model, and drive the bursting neuron populations. For each input level, we ran ten simulation experiments, where we measured the oscillation frequency of the bursting neuron populations and the corresponding swimming speed, as shown in Figure 9. Our results demonstrate that the increase in the tonic excitation input induces a gradual increase in the oscillation frequency and, in turn, an increase in the creature's swimming speed. Interestingly, the oscillation frequency and the swimming speed exhibit a linear relationship across a range, which is typical for undulatory swimming across species [55]. Additionally, the swimming speed saturates at low and high oscillation frequencies. The reason for this is two-fold: First, those frequencies distort the phase lag between consecutive joints, which is necessary for undulatory swimming. Second, at high frequencies, the mechanical properties of the joints and motors do not allow for full-range movements of the joints.

### 8.7 Obstacle Avoidance

The amphibious creature can not only adapt its swimming speed to external inputs, but also adjust its swimming direction to avoid collisions with obstacles in its environment. To demonstrate this, we utilized the distance sensors and our controllers' corresponding feedback network, as described in Section 6. We also chose the undulatory swimming pattern for its higher movement flexibility.

Focusing on static obstacles first, we simulated a complex environment with various objects, as shown in Figure 10A. We initially

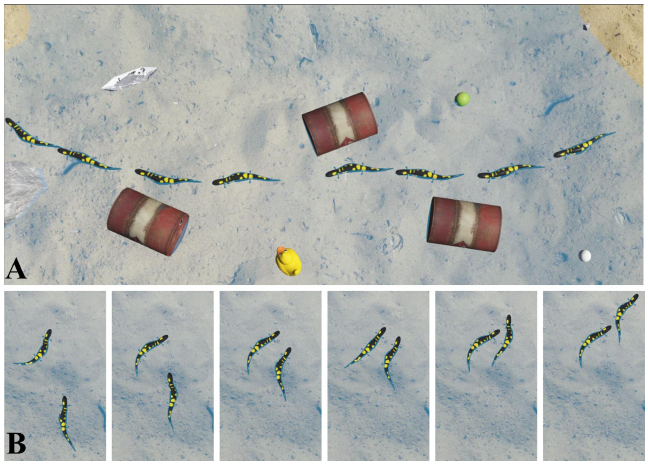


Figure 10: Static (A) and dynamic (B) obstacle avoidance by our closed-loop spiking neural network controllers. The distance sensors detect the presence of obstacles on either side of the creature's head and drive the corresponding sensory neuron populations (see Figures 2A and B). The sensory neurons modulate the spinal contractions to turn the creature from its initially straight trajectory.

oriented the creature such that its straight swimming motion would have driven it into the rightmost barrel and allowed it to swim freely. While initially swimming in a straight line, the feedback from the distance sensors modified the spine contractions and turned the creature to the right. After avoiding the rightmost barrel, its swimming direction pointed towards the middle barrel. Again, the creature sensed the new obstacle and turned to the left to avoid colliding with it. In a similar fashion, it also avoided the left barrel to navigate successfully through the complex environment.

Next, focusing on dynamic obstacles, we introduced an identical swimming creature (left) that was swimming in the pond using the sequential alternating controller, as shown in Figure 10B. We selected this pattern for the left creature because its slower swimming speed allowed for the detection of its thin skeleton by the distance sensors of the right creature. We initially oriented the two swimming creatures so that their straight trajectories would result in a direct collision and allowed them to swim freely. While the one on the right initially swam in a straight line, it modified its spinal contractions as soon as it detected the left creature, and turned right to avoid the collision. In fact, its higher swimming speed due to its undulatory swimming pattern allowed it to swim past the left creature and continue swimming along its updated trajectory.

### 8.8 Method generalization and comparison

Finally, we examined whether our method could be used for the locomotion control of agents beyond the salamander creature and how its performance compared against traditional RL methods. For this, we used the adaptation of our control method presented in Section 7 to drive the locomotion of a quadruped and compared its performance against PPO. Since our SNN controller does not require training due to its biologically inspired architecture, it accumulates rewards right away, while

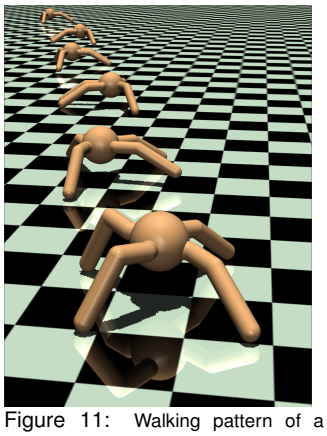

Figure 11: Walking pattern of a quadruped with our proposed SNN.

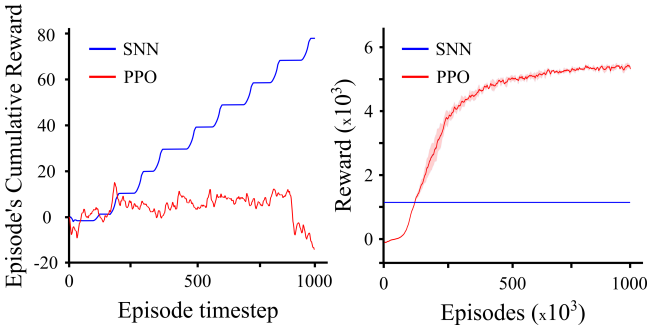


Figure 12: Comparison between the SNN-controlled and the PPO-trained agent. The SNN accumulates higher reward than PPO in the first episodes of training (left). PPO eventually outperforms the SNN controller after 118,337 episodes (right).

PPO performs poorly in the first episodes of its training (see Figure 12). While PPO eventually outperformed our method as expected, it took 118,337 episodes to match and exceed the SNN performance. Moreover, our SNN gave rise to a consistently realistic walking pattern, unlike PPO, which sacrificed motion realism to further increase the covered distance (and, therefore, the rewards). Although the comparison of our method against RL is necessary from an evaluation perspective, it is not perfectly fair. Our control architecture does not require thousands of training episodes to learn the behavior from scratch. However, it requires the domain-expert knowledge of neuroscientists that have identified the biological networks providing these behaviors in animals. In addition, it also requires some manual tuning of hyperparameters to replicate the biological function.

### 8.9 Multi-agent Simulations

Our method scales well and allows for the simulation of multiple amphibious creatures in a complex environment. Specifically, we simulated seven creatures that were driven by our spiking neural network controllers, as shown in Figure 13. The creatures navigated the pond avoiding collisions with each other while moving along their individual trajectories. The creatures would move along a straight line from their initial positions ($T = 0$ s), but their trajectories were modified to avoid collisions with other creatures or due to terrain irregularities. Two of the creatures were initially on the ground (index 1) or in shallow water (index 7) and utilized the crawling pattern to move before transitioning to swimming.

## 9 Limitations

We proposed a bioinspired method for controlling amphibious creatures with multiple DOFs that addresses some drawbacks of well-established methods, such as the long training times and high computational overhead, and also has the potential to reduce power requirements. However, it is also challenged by its own limitations.

While our controller gives rise to three different behaviors, separate subnetworks drive each of them. Although universal controllers are challenging even for mature training methods, the requirement to design a specialized subnetwork for each behavior limits the generality of our approach. Beyond its task-specific design, our architecture is also tailored to the number of DOFs in the creature's skeleton. While extending the spiking neural network to control additional joints in the spine is straightforward, completely different repetitive behaviors require manual intervention. The generalization to tasks for different agents (e.g. biped locomotion) that require balancing based on extensive feedback would introduce additional challenges to our method and require further refinement.

The separation of subnetworks that give rise to the three motion patterns introduces some discontinuity when transitioning between

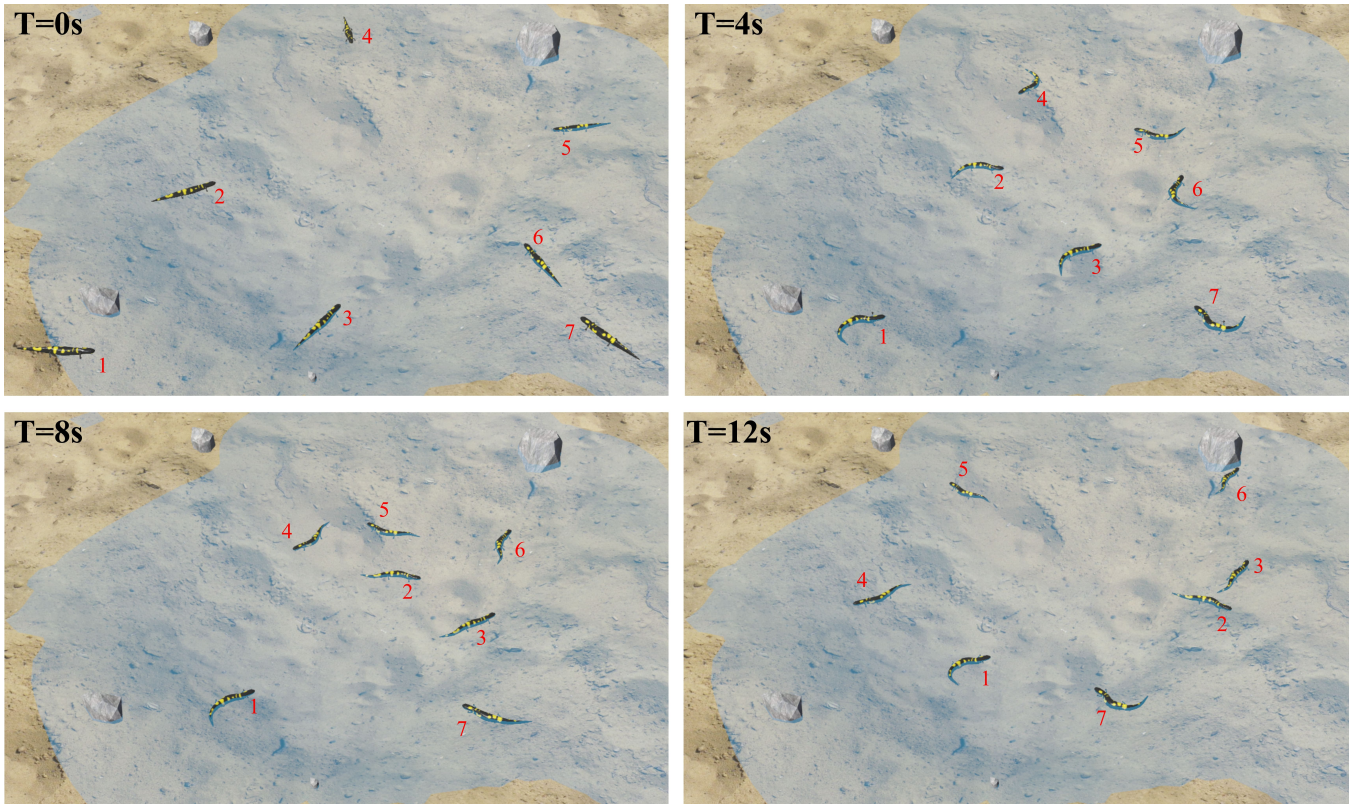


Figure 13: Seven amphibious creatures driven by our spiking neural network controllers interact with each other, successfully avoiding collisions while moving across the pond. Two creatures (1 and 7) initially crawl before transitioning to swimming when they enter the water.

behaviors. Specifically, the leg rotations are synchronized with the alternating contractions of the spine during crawling, as discussed in Section 8.3, providing a smooth transition to the sequentially alternating swimming pattern. However, when transitioning to the undulatory swimming pattern, the transition is abrupt. It would be interesting to investigate control strategies that can allow the creature to smoothly ease in and out between different motion patterns.

Although our method is designed to mitigate the computational requirements of other approaches, computation can still pose as a potential limitation. Our biologically plausible spiking neural network inherits the compact structure of its biological counterpart. However, the size of the neuron populations is decisive for the elegance of the resulting motions. In the current context, we were able to effectively control the amphibious creature with as few as 20 neurons per population, and their total number would scale up linearly with the addition of extra joints. However, increasing the number of neurons per population to 100 or 1000 could make the motion patterns more smooth and intricate (see Figures 1B and C).

Finally, the number of distance sensors and their positioning on the creature's head limited the sensing capabilities of our network. Using a higher number of more sophisticated sensors and integrating them into our feedback loop could further improve the control.

## 10 Conclusion and Future Work

We presented a bioinspired method for controlling an articulated amphibious creature that demonstrated one crawling motion and two swimming motions without the need for any prior training. Our method allows for closed-loop control with feedback from sensory input. This allowed the creature to transition between different motion patterns, avoid collisions with static and dynamic obstacles, and also adaptively control its swimming speed. We demonstrated the scalability of our method by showing multiple amphibious creatures interact with each other in a complex environment.

The structure of the creature on which we tailored our method set the limits for the emergent behaviors that we could target. For example, with the spinal joints moving only along the horizontal plane, we could not target three-dimensional swimming motions or diving and resurfacing behaviors. In the future, we plan to investigate motion control for creatures with more complex skeletons. Moreover, we would like to explore whether and how the inclusion of more sensors of increasing sophistication (inertial units, lidar, etc) would allow for the generalization of our control methods to other tasks and agents (e.g. biped locomotion).

More broadly, it would be interesting to develop a workflow for designing controllers for low-level behaviors that would mitigate the need for domain-expert knowledge and manual design, as mentioned in Section 9. We envision training-free controllers for low-level behaviors, such as locomotion or swimming, being integrated with trained controllers for high-level cognitive tasks for providing efficient performance that matches that of biological agents.

## Acknowledgments

We thank the anonymous reviewers for their valuable feedback. I. P. was supported in part by the Onassis Foundation scholarship. M. A. was supported in part by the Rutgers University start-up grant and the National Science Foundation under awards CCF-2110861 and IIS-2132972. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] C. Allen and C. F. Stevens. An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22):10380–10383, 1994.

[2] M. Ambroise, T. Levi, S. Joucla, B. Yvert, and S. Saïghi. Real-time biomimetic central pattern generators in an fpga for hybrid experiments. *Frontiers in neuroscience*, 7:215, 2013.

[3] E. Angelidis, E. Buchholz, J. Arreguit, A. Rougé, T. Stewart, A. von Arnim, A. Knoll, and A. Ijspeert. A spiking central pattern generator for the control of a simulated lamprey robot running on spinnaker and loihi neuromorphic boards. *Neuromorphic Computing and Engineering*, 1(1):014005, 2021.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[5] Y. Baba, Y. Kake, M. Yoshida, and K. Uematsu. Activities of mesencephalic nucleus neurons during fictive swimming of the carp cyprinus carpio. *Fisheries science*, 69(3):581–588, 2003.

[6] Z. Bing, C. Lemke, L. Cheng, K. Huang, and A. Knoll. Energy-efficient and damage-recovery slithering gait design for a snake-like robot based on reinforcement learning and inverse reinforcement learning. *Neural Networks*, 129:323–333, 2020.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] J. Braure. Participation to the construction of a salamander robot: exploration of the morphological configuration and the locomotion controller. *Biologically Inspired Robotics Group, master thesis*, pp. 1–46, 2004.

[9] T. Bujard, F. Giorgio-Serchi, and G. D. Weymouth. A resonant squid-inspired robot unlocks biological propulsive efficiency. *Science Robotics*, 6(50):eabd2971, 2021.

[10] V. Caggiano, R. Leiras, H. Goñi-Erro, D. Masini, C. Bellardita, J. Bouvier, V. Caldeira, G. Fisone, and O. Kiehn. Midbrain circuits that set locomotor speed and gait selection. *Nature*, 553(7689):455–460, 2018.

[11] S. Coros, P. Beaudoin, and M. van de Panne. Robust task-based control policies for physics-based characters. *ACM Trans. Graph.*, 28(5):1–9, 2009.

[12] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne. Locomotion skills for simulated quadrupeds. *ACM Trans. Graph.*, 30(4), 2011.

[13] A. Crespi, A. Badertscher, A. Guignard, and A. J. Ijspeert. Amphibot i: an amphibious snake-like robot. *Robotics and Autonomous Systems*, 50(4):163–175, 2005.

[14] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[15] 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.

[16] E. Donati, F. Corradi, C. Stefanini, and G. Indiveri. A spiking implementation of the lamprey's central pattern generator in neuromorphic vlsi. In *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, pp. 512–515, 2014.

[17] X. Dong, J. Huang, Y. Yang, and S. Yan. More is less: A more complicated network with less inference complexity. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, p. 251–260, 2001.

[19] J. Fang, C. Jiang, and D. Terzopoulos. Modeling and animating myriapoda: a real-time kinematic/dynamic approach. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 203–212, 2013.

[20] S. Geva and J. Sitte. A cartpole experiment benchmark for trainable controllers. *IEEE Control Systems Magazine*, 13(5):40–51, 1993.

[21] R. Glatt, F. L. Da Silva, and A. H. R. Costa. Towards knowledge transfer in deep reinforcement learning. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 91–96. IEEE, 2016.

[22] S. Glatz, J. Martel, R. Kreiser, N. Qiao, and Y. Sandamirskaya. Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9631–9637. IEEE, 2019.

[23] B. Graf, M. Hans, and R. D. Schraft. Care-o-bot ii—development of a next generation robotic home assistant. *Autonomous robots*, 16(2):193–205, 2004.

[24] J. Gray. How fishes swim. *Scientific American*, 197(2):48–55, 1957.

[25] S. Grillner, Ö. Ekeberg, A. El Manira, A. Lansner, D. Parker, J. Tegner, and P. Wallen. Intrinsic function of a neuronal network—a vertebrate central pattern generator. *Brain Research Reviews*, 26(2-3):184–197, 1998.

[26] D. Gutierrez-Galan, J. P. Dominguez-Morales, F. Perez-Peña, A. Jimenez-Fernandez, and A. Linares-Barranco. Neuropod: a real-time neuromorphic spiking cpg applied to robotics. *Neurocomputing*, 381:10–19, 2020.

[27] S. Huber, R. Poranne, and S. Coros. Designing actuation systems for animatronic figures via globally optimal discrete search. *ACM Trans. Graph.*, 40(4), 2021.

[28] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[29] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[30] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *science*, 315(5817):1416–1420, 2007.

[31] Y. Inagaki, E. Boucher, M. Hong, L. Bouancheau, and F. Nilsson. Cracking the snake code on the bad guys. In *ACM SIGGRAPH 2022 Talks*, SIGGRAPH '22, 2022.

[32] V. Ivanov and P. Havaldar. Artistically directable walk generation. In *ACM SIGGRAPH 2022 Talks*, SIGGRAPH '22, 2022.

[33] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[34] E. Ju, J. Won, J. Lee, B. Choi, J. Noh, and M. G. Choi. Data-driven control of flapping flight. *ACM Transactions on Graphics (TOG)*, 32(5):1–12, 2013.

[35] P. Kormushev, K. Nomoto, F. Dong, and K. Hirota. Time hopping technique for faster reinforcement learning in simulations. *CYBERNETICS AND INFORMATION TECHNOLOGIES*, 11(3), 2011.

[36] R. Krahe and F. Gabbiani. Burst firing in sensory systems. *Nature Reviews Neuroscience*, 5(1):13–23, 2004.

[37] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. In *Computer Graphics and Interactive Techniques*, pp. 155–162, 1996.

[38] Y. Lee, S. Kim, and J. Lee. Data-driven biped control. In *ACM SIGGRAPH 2010 papers*, pp. 1–8. 2010.

[39] A. S. Lele, Y. Fang, J. Ting, and A. Raychowdhury. Learning to walk: bio-mimetic hexapod locomotion via reinforcement-based spiking central pattern generation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):536–545, 2020.

[40] M. A. Lewis, F. Tenore, and R. Etienne-Cummings. Cpg design using inhibitory networks. In *IEEE International Conference on Robotics and Automation*, pp. 3682–3687, 2005.

[41] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao. Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 36(9):2059–2070, 2018.

[42] L. Liu and J. Hodgins. Learning bastsketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[43] C. Mayr, S. Hoeppner, and S. Furber. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *arXiv preprint arXiv:1911.02385*, 2019.

[44] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *ACM SIGGRAPH Computer Graphics*, 24(4):29–38, 1990.

[45] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[46] S. Min, J. Won, S. Lee, J. Park, and J. Lee. Softcon: Simulation and control of soft-bodied animals with biomimetic actuators. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[47] I. Naotunna, C. J. Perera, C. Sandaruwan, R. Gopura, and T. D. Lalitharatne. Meal assistance robots: A review on current status, challenges and future directions. In *IEEE/SICE International Symposium on System Integration*, pp. 211–216, 2015.

[48] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[49] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. on Graph.*, 37(4):1–14, 2018.

[50] I. Polykretis, G. Tang, and K. P. Michmizos. An astrocyte-modulated neuromorphic central pattern generator for hexapod robot locomotion on intel's loihi. In *International Conference on Neuromorphic Systems 2020*, pp. 1–9, 2020.

[51] L. Reveret, L. Favreau, C. Depraz, and M.-P. Cani. Morphable model of quadrupeds skeletons for animating 3d animals. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 135–142, 2005.

[52] J. Rosenblatt, S. Williams, and H. Durrant-Whyte. A behavior-based architecture for autonomous underwater exploration. *Information Sciences*, 145(1-2):69–87, 2002.

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

[54] M. J. Schuster, S. G. Brunner, K. Bussmann, S. Büttner, A. Dömel, M. Hellerer, H. Lehner, P. Lehner, O. Porges, J. Reill, et al. Towards autonomous planetary exploration. *Journal of Intelligent & Robotic Systems*, 93(3):461–494, 2019.

[55] R. E. Shadwick and S. Gemballa. Structure, kinematics, and muscle dynamics in undulatory swimming. *Fish physiology*, 23:241–280, 2005.

[56] T. Shiratori, B. Coley, R. Cham, and J. K. Hodgins. Simulating balance recovery responses to trips based on biomechanical principles. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 37–46, 2009.

[57] W. Si, S.-H. Lee, E. Sifakis, and D. Terzopoulos. Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)*, 34(1):1–15, 2014.

[58] S. Soffe and A. Roberts. Tonic and phasic synaptic input to spinal cord motoneurons during fictive locomotion in frog embryos. *Journal of Neurophysiology*, 48(6):1279–1288, 1982.

[59] S. Spano, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, A. Nannarelli, and M. Re. An efficient hardware implementation of reinforcement learning: The q-learning algorithm. *Ieee Access*, 7:186340–186351, 2019.

[60] R. K. Stagsted, A. Vitale, A. Renner, L. B. Larsen, A. L. Christensen, and Y. Sandamirskaya. Event-based pid controller fully realized in neuromorphic hardware: a one dof study. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10939–10944. IEEE, 2020.

[61] S. Steingrube, M. Timme, F. Wörgötter, and P. Manoonpong. Self-organized adaptation of a simple neural circuit enables complex robot behaviour. *Nature physics*, 6(3):224–230, 2010.

[62] J. Tan, Y. Gu, G. Turk, and C. K. Liu. Articulated swimming creatures. *ACM Trans. Graph.*, 30(4), 2011.

[63] G. Tang, N. Kumar, and K. P. Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6090–6097, 2020.

[64] S. B. Thrun. Efficient exploration in reinforcement learning. 1992.

[65] K. Uematsu and T. Ikeda. The midbrain locomotor region and induced swimming in the carp cyprinus carpio. *Nippon Suisan Gakaishi*, 59(5):783–788, 1993.

[66] Z. Wang, Q. Gao, and H. Zhao. Cpg-inspired locomotion control for a snake robot basing on nonlinear oscillators. *Journal of Intelligent & Robotic Systems*, 85(2):209–227, 2017.

[67] J.-c. Wu and Z. Popović. Realistic modeling of bird flight animations. *ACM Transactions on Graphics*, 22(3):888–895, 2003.

[68] Z. Xie, S. Starke, H. Y. Ling, and M. van de Panne. Learning soccer juggling skills with layer-wise mixture-of-experts. In *ACM SIGGRAPH Conference Proceedings*, 2022.

[69] K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Okada, K. Matsumoto, M. Ishikawa, I. Shimoyama, and M. Inaba. Home-assistant robot for an aging society. *Proceedings of the IEEE*, 100(8):2429–2441, 2012.

[70] B. Yang, L. Han, G. Li, W. Xu, and B. Hu. A modular amphibious snake-like robot: design, modeling and simulation. In *2015 IEEE international conference on robotics and biomimetics (ROBIO)*, pp. 1924–1929. IEEE, 2015.

[71] J. Yu, M. Tan, J. Chen, and J. Zhang. A survey on cpg-inspired control models and system implementation. *IEEE transactions on neural networks and learning systems*, 25(3):441–456, 2013.

[72] B. Zhang, Z. Mao, W. Liu, and J. Liu. Geometric reinforcement learning for path planning of uavs. *Journal of Intelligent & Robotic Systems*, 77(2):391–409, 2015.