Task-Independent Spiking Central Pattern Generator: A Learning-Based Approach

Elie Aljalbout[1,2] · Florian Walter[1] · Florian Röhrbein[1,3] · Alois Knoll[1]

Abstract

Legged locomotion is a challenging task in the field of robotics but a rather simple one in nature. This motivates the use of biological methodologies as solutions to this problem. Central pattern generators are neural networks that are thought to be responsible for locomotion in humans and some animal species. As for robotics, many attempts were made to reproduce such systems and use them for a similar goal. One interesting design model is based on spiking neural networks. This model is the main focus of this work, as its contribution is not limited to engineering but also applicable to neuroscience. This paper introduces a new general framework for building central pattern generators that are task-independent, biologically plausible, and rely on learning methods. The abilities and properties of the presented approach are not only evaluated in simulation but also in a robotic experiment. The results are very promising as the used robot was able to perform stable walking at different speeds and to change speed within the same gait cycle.

Keywords Central pattern generators · Spiking neural networks · Learning · Robotics locomotion · Neurorobotics

✉ Elie Aljalbout
  elie.aljalbout@tum.de

  Florian Walter
  florian.walter@tum.de

  Florian Röhrbein
  florian.roehrbein@in.tum.de

  Alois Knoll
  knoll@in.tum.de

[1] Institut für Informatik VI, Technische Universität München, 85748 Munich, Germany

[2] Present Address: Chair of Robotics Science and Systems Intelligence, Department of Electrical and Computer Engineering, Technische Universität München, 80797 Munich, Germany

[3] Present Address: Alfred Kärcher SE Co. & KG, 71364 Winnenden, Germany

1 Introduction

Robotic locomotion is a functionality that enables robots to navigate and transport themselves. One specific form of locomotion is legged locomotion. Despite the challenges faced to achieve this form, it presents multiple advantages such as adaptability to multiple types of terrains (e.g. unpaved ground, rugged terrain, stairs etc.), interaction with the physical environment, agility in terms of generated paths complexity, and superior failure tolerance (e.g. when a leg fails the robot can still move) [14].

In biology, humans and animals can move and transport their bodies from one place to another in a quite efficient manner, which motivates and highlights the necessity and importance of approaching this problem with more inspiration from biology. In addition, imitating biological locomotion could also lead to more human- and animal-like gaits. However, the functionality leading to biological locomotion is not yet fully understood. Central Pattern Generators (CPGs) are neural networks that are thought to be responsible for multiple rhythmic behaviors, including locomotion. The exact architecture and configuration leading to the ability of these networks to produce rhythmic patterns are still not entirely known. Consequently, this field of research is considered multidisciplinary, since it contributes to both robotics and neuroscience. Another advantage of engineering CPGs is the possibility to implement them on neuromorphic hardware [23], leading to more power efficiency.

In robotics, many attempts were made to design CPGs for locomotion control. These attempts can be separated into two main groups:

1. Methods based on systems of coupled oscillators (SCO)
2. Spiking neural networks (SNN) based methods

The second type is the one adopted in this work. As for SCOs, they offer the ability to be mathematically analyzed due to their mathematical nature, which made them advance faster than their counterparts that present complex dynamics and harder analytical obstacles. Unfortunately, SCOs lack biological plausibility, which means that their contributions to neuroscience and interventions in neuro-prosthetics are limited. For example, when interfacing prosthetic robotic devices to amputated humans and spinal cord injury patients, SNN-based approaches process and receive the same kind of signals as the CPGs of their biological counterparts [7]. The same doesn't apply for SCOs.

In this paper, we introduce a novel SNN-based CPG architecture that combines biological plausibility with easy tunable output behavior. The architecture builds on previously published models but adds three notable features: First, the gait speed can be easily modulated through a tonic input signal even within an ongoing cycle. Second, the generated patterns are not encoded manually or through evolutionary algorithms but directly learned based on a target signal. Third, the number of phases can be defined freely, which, together with the second feature, allows the approach to be general and task-independent. As we will show during the evaluation, this approach is very stable and converges quickly, making it very well-suited for implementing it in robots. Importantly, while also previously-proposed CPG architectures support speed modulation, pattern learning, or flexible number of phases, none of them supports all of these properties at the same time. Our contribution therefore can both provide valuable insight to both neuroscience-focused CPG research and robot locomotion research.

The rest of this paper is structured as follows. In Sect. 2, related methods will be presented, followed by the proposed approach in Sect. 3. Results and conclusion will be then discussed in Sects. 4 and 5 respectively.

2 Related Work

Several approaches have been demonstrated to be capable of building CPGs using spiking neural networks. Some of these methods are simply based on hand-tuning of the synaptic efficacies [3,5,8,12,13,15]. Such methods manage to reproduce basic CPG functionalities such as stable walking and in some cases speed control [13,15]. The main disadvantage of these approaches is the small size of their networks[1] which makes the weight tuning process easier but forbids the network from scaling to additional features such as modulation by sensory feedback and autonomous gait transition. Another set of approaches relies on genetic algorithms and similar methods to learn the network weights. For instance, the work in [20] applied genetic evolution on the network from [13] and succeeded to obtain a similar behavior, omitting the complicated tuning of the locomotion behavior but keeping the same disadvantages as previously discussed. An alternative approach was recently presented in [6,7], where the CPG network is formulated as a Christiansen grammar and an evolutionary process is performed on it. Interestingly, this method not only learns the network's weights but also its architecture, which is a great advantage offering more automation for the CPG building process. Further approaches based on evolutionary algorithms were also proposed in [17] and [4]. Besides the obvious disadvantage of slow convergence, evolutionary methods don't scale to larger networks and their training method is not biologically plausible, which also limits their neuroscientific contribution. Other methods also fall in the category of non-plausible to biology such as [22], where it is proposed to perform brute-force search on the CPG parameters space, and [11] where a gradient descent based method is presented. These methods have multiple disadvantages such as slow convergence (especially in the case of brute-force) and being incapable to scale to advanced CPG features. Finally, the method in [19] presents a general framework to build task-independent CPGs. It relies on the remote supervision method (ReSuMe) [18] to learn the weights connecting two populations of randomly connected neurons to the motor neurons of the CPG. This method uses biologically plausible learning but a non-plausible network architecture. Additionally, it only achieves stable walking at a fixed speed and omits additional CPG features.

3 Proposed Approach

As previous methods either relied on hand-tuning, neglected advanced CPG features, or couldn't scale to larger CPG networks, we present the following approach, which is the first to successfully combine all of these characteristics using a biologically plausible network architecture. From a general perspective, the architecture consists of three main components. The first one is the neural phase generator (NPG), which is adapted from [15] with minor modifications. In analogy to the biological model in [21], this part plays the role of the rhythm generator. The second part is the pattern forming network (PFN), which consists of networks with random parameters. At this level spiking patterns appear in a rhythmic fashion with respect to the NPG's output. Finally, the last part is the pool of motor neurons, which activates the actuated muscles through spike activity. Each of these components will be discussed next in more details.

Figure 1 shows the proposed architecture in the case of a CPG which encodes only two phases. For this reason, the architecture is almost symmetric, when omitting the randomness

---

[1] CPGs are known to have a small number of neurons relative to the whole nervous system. The discussed methods use networks that are far smaller than biological CPGs.
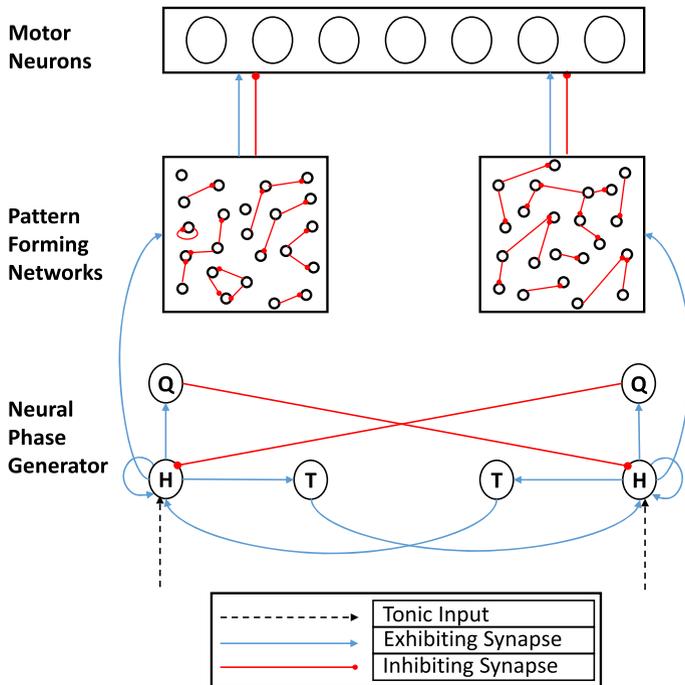
Fig. 1 The proposed network architecture for a CPG with two phases. The neural phase generator (NPG) generates a rhythm corresponding to the number of desired phases. The H neuron is the representative of the activity of each phase, Q takes care of inhibiting other phase modules and T is responsible for the transition to another phase. This rhythm is then propagated to the pattern forming networks (PFN) which map the NPG outputs to a higher dimensional space. PFNs output is then fed to motor neurons which activate the robot's muscles

in the PFNs. In the case when more phases are required, the architecture is extended in all three parts. For instance, at the NPG level, more modules should be added corresponding to the new requirement (Fig. 2). Similarly, at the second level, random networks are also created with respect to the number of desired phases. Finally, in the pool of motor neurons, the number of neurons can also be different but not necessarily. In the following, each of these components will be described separately, in terms of their inner connections, functionality, and role.

3.1 Neural Phase Generator

As already mentioned, this part is adapted from [15]. Briefly, each NPG module consists of three neurons H, Q and T (as seen in Fig. 2). The H neuron is the representative of the module's activity. In other words, whenever a module's H neuron is spiking, the phase this module represents is active. As for the Q neuron, its role is to guarantee that other NPG modules are inactive when required. It does so by inhibiting the H neurons of the corresponding modules. As for T neurons, they are responsible for the transition from module to module. In contrast to the NPG in [15], our modified version receives an external tonic input, which is responsible for the start and end of activity by exciting and inhibiting the neurons of the NPG. This input represents the modulating inputs issued from the brain to control the CPG's activity.
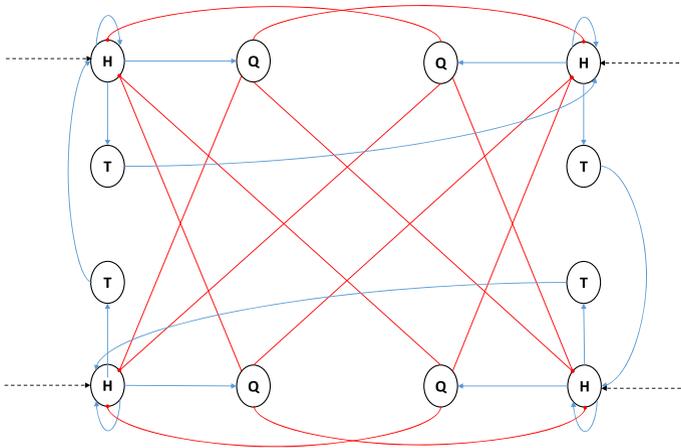
Fig. 2 Illustration of an NPG network for the case of a CPG having four phases. Blue and red arrows represent exhibiting and inhibiting synapses respectively. (Color figure online)

Once the NPG is active, cutting the external input won't stop its activity. Instead, the network will keep oscillating because of its inner mutual connections. This behavior is compatible with the nature of CPGs being able to oscillate even in the absence of input from the brain. To make sure this property is available, a constraint is required: the auto-synapse of the H neurons should have weights that are high enough to ensure a continuous spiking behavior until the respective T neurons fire, which then results in a phase transition. Moreover, when the tonic input has a higher frequency, the H neuron will fire more frequently resulting in a faster transition to the next phase. This modification gives control to the NPG over the speed and phase properties of the produced gaits. This behavior is plausible with the proposed biological model where this control is available at the level of the rhythm generator. In [15], this ability is given to the second layer of the architecture which in their case is the motor output shaping stage. Compared to our architecture this would correspond to giving control to the PFNs.

Another modification, to the NPG of [15], is in the number of T neurons in each module. In the previous work, two T neurons were present in every module, as for ours, this number is variable. It is dependent on the duration of the phase each module is responsible for, the longer the duration the bigger the number. All T neurons except the last one, receive the same external tonic input (as the H neuron), which is important since they each excite a part of the PFN. However, these intermediate T neurons, in addition to their tonic inputs are not visible in Fig. 2 for the sake of simplicity.

Finally, the number of modules in our version of the NPG is not fixed to two, but instead is flexible depending on the number of desired phases, as illustrated in Fig. 2. This flexibility allows the architecture to be general and adaptable for different types of locomotion.

3.2 Pattern Forming Networks (PFN)

This part of the architecture consists of several separate networks. The number is equivalent to the number of phases of the desired CPG. Each of these networks corresponds to a module of the NPG and is excited by its H and intermediate T neurons. The neurons within each of these networks have randomly initialized properties, leading to richer dynamics and higher learning abilities.

Fig. 3 Illustration of inhibiting
networks. Neurons in the left
column are the inhibiting
neurons, and the ones on the right
are those of the PFN. Blue and
red arrows represent exhibiting
and inhibiting synapses
respectively. (Color figure online)
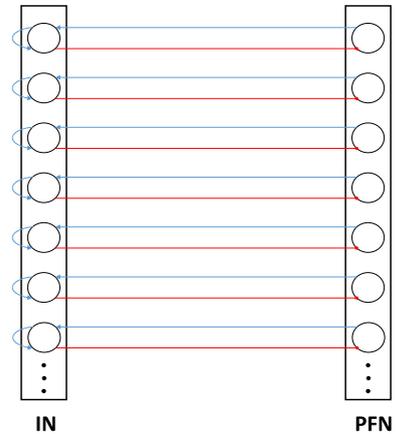
In order to simplify the learning procedure, a minor restriction is enforced: neurons of the PFNs are required to spike only once during each CPG cycle. This is achieved by creating another network with a similar number of neurons for every PFN. This second network is named inhibiting network (IN). Both networks are then connected in a one-to-one fashion with excitation issued from the PFN to the IN and with inhibition in the opposite direction. INs have no synapses connecting their neurons, but instead, exciting auto-synapses for all of their neurons to ensure that each one of them keeps spiking once it's activated. To illustrate how this works: each time a neuron in a PFN spikes, it'll excite a corresponding neuron in an IN leading it to spike in a continuous fashion and inhibit reciprocally the same PFN neuron that made it fire. This concept is illustrated in Fig. 3. Now to make sure that the IN is reset in the next cycle, it'll be inhibited by a Q neuron of an NPG module from a different phase.

Additionally, neurons within a certain PFN are connected with random mutual inhibition at a 10% rate of connectivity (i.e. the number of synapses is 10% the one of an all-to-all connectivity scheme), which ensures that spikes produced by different neurons within the network are well distributed over the whole duration of the corresponding phase, and not concentrated in a small duration right after the NPG spikes. Distribution of the PFN spikes in time is very important for the learning, in a way that the more spikes are distributed, the more the number of desired behaviors that can be learned increases. The intuition behind that is simple. To illustrate it, consider the case where no spike occurs in the time interval $[t, t + \epsilon]$ at the PFN. This will make it less probable that any spike occurs in this same interval in the pool of motor neurons, which are only excited by PFN neurons. In conclusion, PFN spikes should be the most possibly scattered over the phase duration. To enforce that, the size of the network can be increased in order to obtain a higher spiking rate. Conceptually, the role of the PFN can be seen as mapping the NPG spikes into a higher dimensional space in which spikes are well distributed in time. Finally, the external control inputs of the NPG are also propagated to this layer. When the frequency of these signals is increased, the frequency of spiking in the H and intermediate T neurons in the NPG will also increase. Thus, the neurons of the PFN will fire sooner, which means that also the speed of gait will increase. The relation between the frequency of spiking at the NPG level and the time of spikes in the PFN is not a perfect mapping, but that is left for the learning to figure out. Concerning the learning, it is important to note that it's dependent on the PFN's size. For instance, the larger the PFN is, the more probable it is that the learning method will converge. However, the convergence will then require longer training time. Throughout our experiments, we used PFNs with sizes
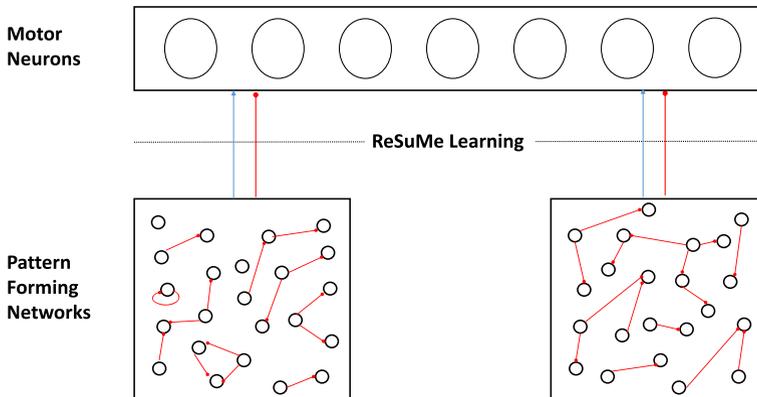
Fig. 4 Pool of motor neurons, which receive excitatory and inhibitory inputs from the PFN. The weights of the synapses connecting PFNs and these neurons are learned using the remote supervision method ReSuMe. Blue and red arrows represent exhibiting and inhibiting synapses respectively. (Color figure online)

ranging from 150 to 500 neurons. The exact selection of the number is also dependent on each phase's duration.

3.3 Pool of Motor Neurons

This last part of the architecture is where rhythmic patterns really matter, since the spikes at this level are responsible for muscle activation and producing gaits for legged locomotion. In biology, the precise way these spikes are decoded into muscle activation is not fully known, but research has indicated that population rate coding is used [9,16]. In other terms, the activation of a muscle is encoded by a population of neurons. Their spike rate is related to the level of activation of the muscle. Therefore, the number of neurons in the pool is dependent on the desired locomotion and the number of actuated muscles. A similar scheme was adopted in the robotic experiment performed within this work. Moreover, neurons within the pool are not inter-connected, but receive as input the spikes produced by the PFNs, through synapses of both excitatory and inhibitory types. The weights of those synapses are the only ones that are learned (Fig. 4). They are adapted to obtain the desired locomotion behavior. The learning procedure will be discussed later. Finally, the control signals (external tonic input) applied at the NPG are also propagated to the neurons in this pool. For instance, when the tonic input has an increased frequency, the speed of the spike pattern produced by the PFN will increase. Ideally, this will lead to the spiking behavior of the motor neurons to be accelerated.

3.4 Learning

When designed faithfully to the constraints already mentioned, the previously described architecture is capable by itself of producing rhythmic spiking patterns in the pool of motor neurons. However, this rhythmic spiking behavior is random and does not correspond to any locomotive behavior. Therefore, the network should be trained somehow to adapt to the desired spiking behavior[2] in order for it to be used in real robot locomotion tasks. In order to do so, the Remote Supervision Method (ReSuMe) [18], was employed. However, the only

---

[2] The desired spiking behavior is specified by the CPG designer and can also be given by demonstration.

synapses of the network which learning is applied on, are the ones connecting the PFNs to the pool of motor neurons.

The way ReSuMe works is as follows [18]: For each of the learning neurons (in our case the motor neurons), a teacher signal is associated with predetermined timing, representing the desired spiking behavior. This signal is not delivered to the learning neuron, but still plays an important role in the learning updates of synapses terminating at it. Moreover, the modification is based on two rules, the first one depends on the correlation between the presynaptic and the desired spike times, as for the second, it depends on the correlation between the presynaptic and the postsynaptic spike times [18]. The following is the corresponding modification function for the synaptic weight between a presynaptic neuron k (from the PFN) and a postsynaptic neuron i (from the motor neurons):

$$
\begin{aligned}
\frac{d}{dt} w_{ki}(t) = S^d(t)\Big[a^d + \int_0^\infty W^d(s^d)S^{in}(t - s^d)ds^d\Big] \\
+ S^l(t)\Big[a^l + \int_0^\infty W^l(s^l)S^{in}(t - s^l)ds^l\Big]
\end{aligned}
\tag{1}
$$

where S^d, S^l and S^{in} are respectively the target, the post and the presynaptic spike trains, a^d and a^l determine the so-called non-Hebbian processes of weight modifications, and s^l and s^d represent respectively the difference between the time of spike of the postsynaptic neuron and the presynaptic one and the difference between the time of the teacher signal and the time of spike of the presynaptic neuron. In the case of excitatory synapses, the terms a^d and a^l are positive, and negative otherwise. As for W^d and W^l, they represent the learning windows, and are formulated as follows:

$$
W^d(s^d) = \begin{cases} +A^d \cdot exp\left(\frac{-s^d}{\tau^d}\right), & \text{if } s^d > 0 \\ 0, & \text{if } s^d \le 0 \end{cases}
\tag{2}
$$

$$
W^l(s^l) = \begin{cases} -A^l \cdot exp\left(\frac{-s^l}{\tau^l}\right), & \text{if } s^l > 0 \\ 0, & \text{if } s^l \le 0 \end{cases}
\tag{3}
$$

where W^d and W^l correspond respectively to the learning window of the target and postsynaptic neurons. As for A^d, A^l, τ^d and τ^l, they are all constants, such that A^d, A^l are positive for excitatory synapses and negative otherwise and τ^d, τ^l are always positive. When setting $a^d = -a^l = a$, $\tau^d = \tau^l$ and $A^d = A^l$, then equation (1) takes the following form:

$$
\frac{d}{dt} w_{ki}(t) = [S^d(t) - S^l(t)]\Big[a^d + \int_0^\infty W^d(s)S^{in}(t - s^d)ds^d\Big]
\tag{4}
$$

Intuitively, what the method does to the network is to strengthen (weaken) synapses which are excitatory (inhibitory) and are incoming to a certain motor neuron, when a spike is transmitted through these synapses within the learning window as defined by τ, whenever the subject motor neuron is desired to spike. This rule pushes the motor neurons to spike at the desired times. Additionally, it weakens (strengthens) synapses which are excitatory (inhibitory) and are incoming to a certain motor neuron when a spike is transmitted through these synapses within the learning window defined by τ, whenever the motor neuron actually spikes. As for this rule, it forbids motor neurons from spiking at undesired times.

Table 1 Parameters of the spiking neurons used in the experiments

Properties Neurons	External input current (pA)		Resting membrane potential (mV)		Capacity of the membrane (pF)		Membrane time constant (ms)		Spike threshold (mV)		Reset potential (mV)	
	min	max	min	max	min	max	min	max	min	max	min	max
CPG	0.0		-70.0		250.0		10.0		-55.0		-70.0	
PFN	0.0	150.0	-90.0	-70.0	100.0	300.0	9.0	30.0	-50.0	-30.0	-90.0	-60.0
Motor	0.0		-70.0		250.0		10.0		-55.0		-70.0	

4 Experiments

We designed experiments to answer the following questions:

– To which extent is the presented approach capable of learning desired spiking patterns?
– Is the targeted speed modulation property working after the learning procedure is done?
– Can the system control the locomotion of legged robots?

For the first two questions, we designed experiments in a simulation-only environment as described in Sect. 4.2. Additionally, we built a real-world experiment with a legged robot (Sect. 4.3) to answer the last question and to validate the second one again. The setup for both experiments is described in Sect. 4.1.

4.1 Setup

In order to allow our results to be reproducible, we provide the implementation details we used along with our experiments. Namely, we used the NEST simulator [10]. In all of our experiments, the leaky integrate-and-fire neuron model [1] was used with alpha-function shaped synaptic currents. As shown in Table 1, all neurons used the NEST default parameters except the ones of the PFN, which are initialized with random uniform values. Parameters that are not mentioned in Table 1 are set to default. As for synapses, they had different initialization ranges at different levels. At the PFN level, neurons were connected to each other with synapses initialized with uniformly random weights between -3 and -1. Neurons of the PFN and motor neurons are connected together with an all-to-all connectivity pattern. We experimented with several numbers of PFN neurons between 300 and 3000 and all values worked out. It is important to note that a larger number of neurons allows a smaller spike shift error, however leads to a longer training time. 20% of the total number of neurons at this level are inhibitory and have initial weights initialized with uniform random values between -25 and -1. The rest of the neurons are excitatory and have their weights initialized with uniform random values between 1 and 5.

As for the learning, we tried several combinations of hyper-parameters for ReSuMe and report the following ones as being the best in terms of speed and accuracy of learning. For learning windows W^d and W^l we used $3ms$. Additionally we found out that $a = 3$, $A^d = A^l = 6$ and $\tau_d = \tau_l = 2$ perform best.

4.2 Simulated Experiments

In order to initially validate the proposed method, we tested it in simulation. Namely, we applied the method to more than 250 randomly generated target spiking sequences with different numbers of motor neurons and let the network learn to imitate these sequences. We
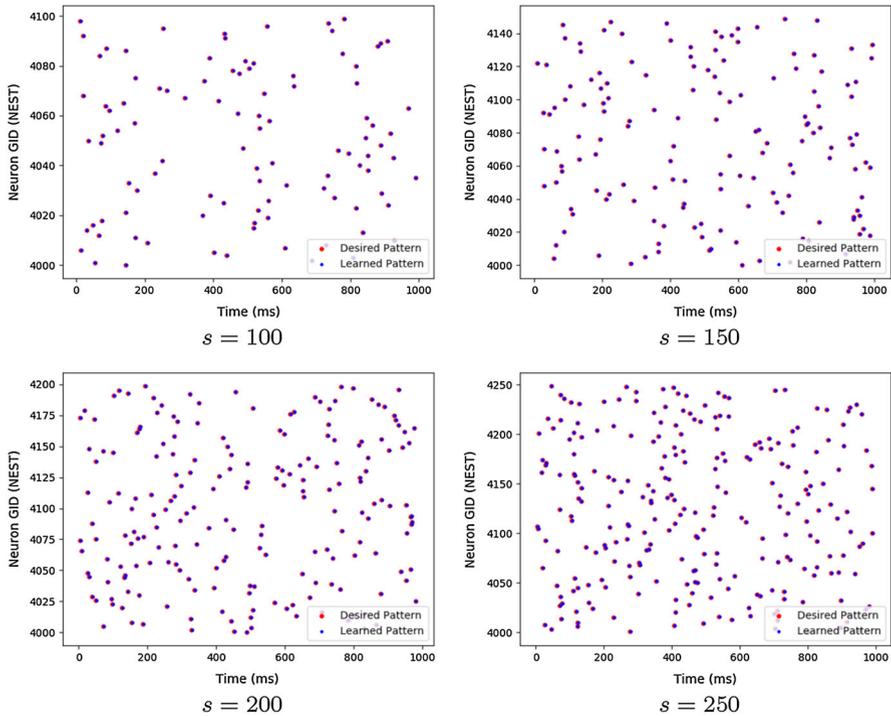
Fig. 5 Spiking patterns obtained for CPGs with s motor neurons for a single CPG cycle. Red dots represent desired spiking patterns randomly generated during the testing phase. Blue dots represent the patterns learned using the presented method. (Color figure online)

experimented with pools of motor neurons of sizes ranging from 5 to 250. As a result, the method managed to successfully learn how to imitate all presented spiking sequences without any need for parameter hand-tuning. The average spike shift error of all experiments was very low ($\sim 2.1ms$). Figure 5 shows four different spiking patterns that have been successfully learned at the level of the motor neurons. These patterns are generated by our CPG within a single cycle. These results show that the proposed architecture together with ReSuMe enables learning any desired locomotive behavior with minimal design effort. Consequently, it is possible to embed any desired locomotive gait (whether measured from an already existing walking system or self-defined) in a spiking neural network.

Additionally, the obtained CPGs are capable of acceleration and deceleration of the same spiking sequences when presented with a higher and lower frequency tonic inputs respectively. Figure 6 illustrates this property. In the first row, a CPG with a small number of motor neurons is shown, as it's easier to watch the modulation property with a sparse plot. However, our method is also capable of modulating larger numbers of motor neurons as shown in the second row. In most cases, we obtained such results by training the CPG network using only one tonic input frequency. However, in some rare cases, speed modulation was only obtained after training the network under different tonic input frequencies.
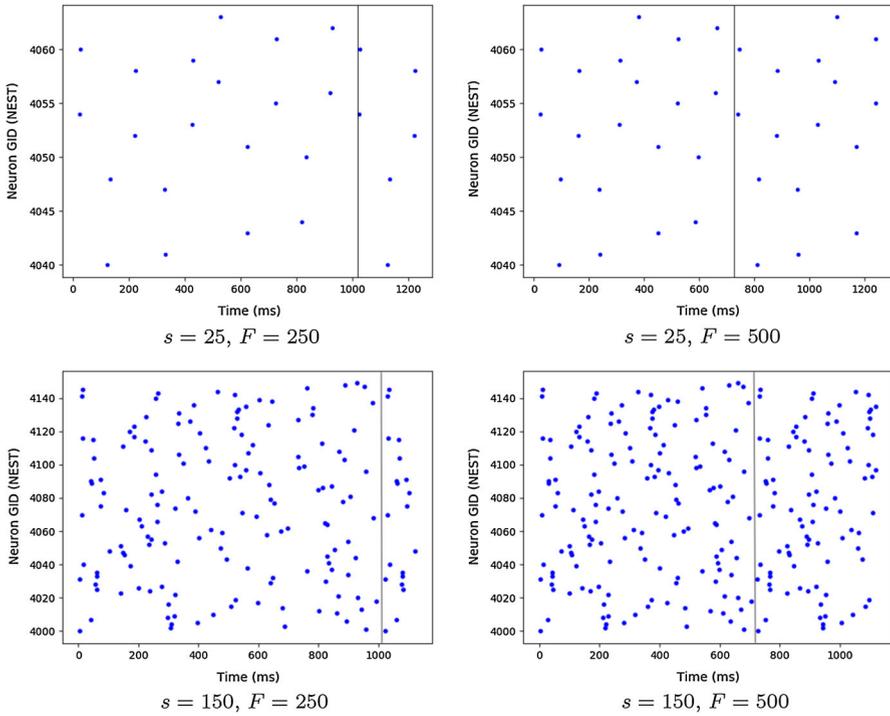
Fig. 6 Speed modulation of two CPGs with different number of motor neurons s in each row. When increasing the spiking frequency F of the tonic input from 250 spikes/s to 500 spikes/s we obtain an increased speed in the motor output. Vertical lines indicate the end of a spiking cycle
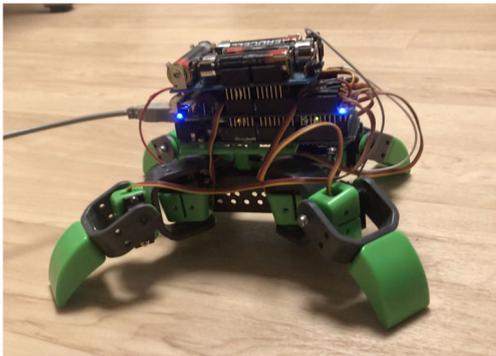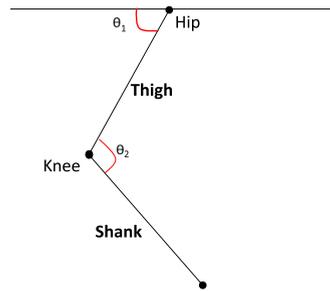

(a)   **(b)**

Fig. 7 **a** ALLBOT VR408. **b** Joints of a single robot leg

4.3 Robotics Experiment

We tested our approach on a robot (Allbot VR408 [2] shown in Fig. 7). The Allbot is a simple quadruped robot with four legs and eight joints that are actuated by standard servo motors. It is equipped with an Arduino Mega board that is used in the experiment to relay

(a) $F = 250\ spikes/s$ **(b)** $F = 500\ spikes/s$
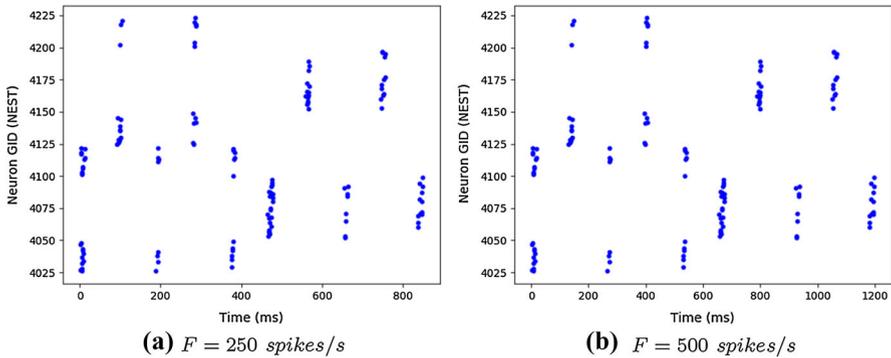
Fig. 8 Outputs of the motor neurons used in the robot experiment under different tonic input frequencies F

motion commands from the network simulation running on the host computer to the servos. The outputs of the motor neurons were used to encode the joint angles. Namely, we used population rate coding, such that the conversion from spikes to angles is done based on:

$$\theta_i = \frac{n_i}{N_i} * \pi \tag{5}$$

where θ_i is the angle of the joint i, n_i is the number of spikes of the motor neurons of the joint i, and N_i is the total number of neurons representing the ith joint. Each joint is represented by a set of 25 neurons. Activations within a small window of time[3] correspond to the angle command. To generate the desired spiking pattern, we used Eq. (5) to obtain the number of spikes for each joint n_i, and generated n_i spikes at different randomly chosen neurons encoding joint i. The motor activations obtained in this experiment are illustrated in Fig. 8. Similarly, other control types (such as joint velocity control) can be applied. After learning a predefined motor spiking behavior, the robot is capable of stable walking at different speeds corresponding to different tonic inputs. We provide a video of this experiment[4]. The code for all experiments will be made available upon publication of this paper.

5 Conclusion

Spiking CPGs are promising techniques for robot locomotion. Previous methods have either missed the generality feature, scalability, biological plausibility or required parameter hand-tuning. In this work, we were able to present the first general framework for building spiking CPGs that includes all of the desired features. Namely, the framework allows to build CPGs for different types of robots with a minimal design effort, and that is done using a learning algorithm. The used learning method is the remote supervision method which was chosen for its fit to this work's needs and its proved convergence and biological plausibility. Additionally, the resulting CPGs have architectures, behaviors, and features similar to biological counterparts. For instance, they can be modulated by external signals to change the speed of walking even during the process of locomotion. We also showed how our CPG can success-fully learn a target spiking behavior allowing it to achieve desired gaits on target robots. A

---

[3] The time window of spike decoding is dependent on the control frequency of the robot.

[4] https://sites.google.com/view/task-independent-cpg/home.

possible extension of this work would be to use reinforcement learning to learn the weights of the network, eliminating the need for behavior demonstration.

References

1. Abbott LF (1999) Lapicque's introduction of the integrate-and-fire model neuron (1907). Brain Res Bull 50(5–6):303–304
2. A robotic toolkit for building quadruped and biped robots, consists of 3D printing and servo motors. http://www.allbot.eu. Accessed 15 Oct 2017
3. Billard A, Ijspeert AJ (2000) Biologically inspired neural controllers for motor control in a quadruped robot. In: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks, vol. 6, pp 637–641. IEEE
4. Caamano P, Becerra JA, Bellas F, Duro RJ (2008) Using spiking neural networks for the generation of coordinated action sequences in robots. In: International conference on neural information processing, pp 1013–1020. Springer
5. Donati E, Corradi F, Stefanini C, Indiveri G (2014) A spiking implementation of the lamprey's central pattern generator in neuromorphic VLSI. In: Biomedical circuits and systems conference, pp 512–515. IEEE
6. Espinal A, Rostro-Gonzalez H, Carpio M, Guerra-Hernandez EI, Ornelas-Rodriguez M, Puga-Soberanes HJ, Sotelo-Figueroa M, Melin P (2016) Quadrupedal robot locomotion: a biologically inspired approach and its hardware implementation. Comput Intell Neurosci. https://doi.org/10.1155/2016/5615618
7. Espinal A, Rostro-Gonzalez H, Carpio M, Guerra-Hernandez EI, Ornelas-Rodriguez M, Sotelo-Figueroa M (2016) Design of spiking central pattern generators for multiple locomotion gaits in hexapod robots by christiansen grammar evolution. Front Neurorobot. https://doi.org/10.3389/fnbot.2016.00006
8. Garcia-Saura C (2015) Central pattern generators for the control of robotic systems. ArXiv preprint arXiv:1509.02417
9. Gerstner W, Kreiter AK, Markram H, Herz AV (1997) Neural codes: firing rates and beyond. Proc Nat Acad Sci 94(24):12740–12741
10. Gewaltig MO, Diesmann M (2007) Nest (neural simulation tool). Scholarpedia 2(4):1430
11. Kuroe Y, Iima H (2006) A learning method for synthesizing spiking neural oscillators. In:International joint conference on neural networks, pp 3882–3886. IEEE
12. Lewis MA, Hartmann MJ, Etienne-Cummings R, Cohen AH (2001) Control of a robot leg with an adaptive avlsi CPG chip. Neurocomputing 38:1409–1421
13. Lewis MA, Tenore F, Etienne-Cummings R (2005) CPG design using inhibitory networks. In: Proceedings of the 2005 IEEE international conference on robotics and automation, pp 3682–3687. IEEE
14. Machado T, Silva MF (2006) An overview of legged robots. In: International symposium on mathematical methods in engineering. MME Press Ankara, Turkey
15. Maufroy C, Kimura H, Takase K (2008) Towards a general neural controller for quadrupedal locomotion. Neural Netw 21(4):667–681
16. Maunsell JH, Van Essen DC (1983) Functional properties of neurons in middle temporal visual area of the macaque monkey. i. selectivity for stimulus direction, speed, and orientation. J Neurophysiol 49(5):1127–1147

17. Orchard G, Russell A, Mazurek K, Tenore F, Etienne-Cummings R (2008) Configuring silicon neural networks using genetic algorithms. In: international symposium on circuits and systems, pp 1048–1051. IEEE
18. Ponulak F (2005) ReSuMe - new supervised learning method for Spiking Neural Networks. Technical Report, Institute of Control and Information Engineering, Poznan University of Technology
19. Ponulak F, Belter D, Kasinski A (2006) Adaptive central pattern generator based on spiking neural networks. In: Proceedings of EPFL LATSIS symposium 2006, dynamical principles for neuroscience and intelligent biomimetic devices, pp 121–122
20. Russell A, Orchard G, Etienne-Cummings R (2007) Configuring of spiking central pattern generator networks for bipedal walking using genetic algorithms. In: International synopsium on circuits and systems, pp 1525–1528. IEEE
21. Rybak IA, Shevtsova NA, Lafreniere-Roula M, McCrea DA (2006) Modelling spinal circuitry involved in locomotor pattern generation: insights from deletions during fictive locomotion. J physiol 577(2):617–639
22. Smolinski TG (2015) Automated parameter search in small network central pattern generators. Springer, New York, pp 292–294. https://doi.org/10.1007/978-1-4614-6675-823
23. Walter F, Röhrbein F, Knoll A (2015) Neuromorphic implementations of neurobiological learning algorithms for spiking neural networks. Neural Netw 72:152–167