Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 5615618, 13 pages
http://dx.doi.org/10.1155/2016/5615618

Research Article

Quadrupedal Robot Locomotion: A Biologically Inspired Approach and Its Hardware Implementation

A. Espinal,[1] H. Rostro-Gonzalez,[2] M. Carpio,[1] E. I. Guerra-Hernandez,[2] M. Ornelas-Rodriguez,[1] H. J. Puga-Soberanes,[1] M. A. Sotelo-Figueroa,[3] and P. Melin[4]

[1]*Division of Postgraduate Studies and Research, Leon Institute of Technology, 37290 Leon, GTO, Mexico*
[2]*Department of Electronics, DICIS, University of Guanajuato, 36885 Salamanca, GTO, Mexico*
[3]*Department of Organizational Studies, DCEA, University of Guanajuato, 36250 Guanajuato, GTO, Mexico*
[4]*Division of Postgraduate Studies and Research, Tijuana Institute of Technology, 22414 Tijuana, BC, Mexico*

Correspondence should be addressed to H. Rostro-Gonzalez; hrostrog@ugto.mx

Received 12 February 2016; Revised 6 April 2016; Accepted 24 May 2016

Academic Editor: Ricardo Aler

A bioinspired locomotion system for a quadruped robot is presented. Locomotion is achieved by a spiking neural network (SNN) that acts as a Central Pattern Generator (CPG) producing different locomotion patterns represented by their raster plots. To generate these patterns, the SNN is configured with specific parameters (synaptic weights and topologies), which were estimated by a metaheuristic method based on Christiansen Grammar Evolution (CGE). The system has been implemented and validated on two robot platforms; firstly, we tested our system on a quadruped robot and, secondly, on a hexapod one. In this last one, we simulated the case where two legs of the hexapod were amputated and its locomotion mechanism has been changed. For the quadruped robot, the control is performed by the spiking neural network implemented on an Arduino board with 35% of resource usage. In the hexapod robot, we used Spartan 6 FPGA board with only 3% of resource usage. Numerical results show the effectiveness of the proposed system in both cases.

1. Introduction

Autonomous robot locomotion is a problem that has been partially solved. To deal with the inherent locomotion problems, different mechanisms have been implemented. Some of them have been implemented through wheels due to simplicity. However, this approach presents disadvantages related to the environment surface. Hence, previous research work has addressed the feasibility of implementing locomotion mechanism based on legs. Legged locomotion results from complex, high-dimensional, nonlinear, dynamically coupled interactions between an organism and its environment. Fortunately, models have been proposed to resolve the redundancy of multiple legs, joints, and muscles by seeking synergies and symmetries [1]. In the literature, there are two main approaches for the design of locomotion control systems such as kinematic and dynamic mathematical models and biologically inspired approaches [2]. In the first one, to move a leg in a desired trajectory, the joint angles are calculated in advance, by using a mathematical model that incorporates both robot and environment parameters, to produce a sequence of actions algorithmically scheduled [3]; these kinds of algorithms can be too complex and unable to be used in dynamic environments with real time response restrictions. The second approach uses bioinspired principles found in nature. From an engineering viewpoint, the main reason for the great interest in bioinspired approaches is the fact that they provide suitable solutions for the design of efficient walking robots. Usually, bioinspired solutions use common principles found in a large variety of animals. Applications of these principles are possible since major advances have been made by biologists in understanding animal locomotion, and at the same time artificial locomotion systems are interesting topics of study, in particular robotics, since they are a good realistic way to verify a hypothesis regarding the biological model and a good source for new ideas [4].

Biologists often assume that vertebrate locomotion is controlled by a Central Pattern Generator (CPG) capable of producing rhythmic patterns or gaits. CPGs have the ability to automatically generate complex control signals for the coordination of muscles during rhythmic movements, such as walking, running, swimming, and flying [5, 6]. CPGs have been used to control a variety of different types of robots and different modes of locomotion. For example, CPG models have been used with hexapod and octopod robots inspired by insect locomotion, quadruped robots inspired by vertebrates, such as horse, biped robots inspired by humans, and other kinds of robots inspired by reptiles, such as snakes. Different levels of abstraction have been used to model CPGs; depending on the phenomena under study the CPG can be designed from detailed biophysical models to abstract systems of coupled oscillators [2, 7]. CPGs present several interesting properties including distributed control, the ability to deal with redundancies, robustness against perturbations, and feedback loops allowing modulation of locomotion in unknown environments by simple control signal. These properties, when transferred to mathematical models, make CPGs interesting building blocks for locomotion controllers in mobile robots [7].

CPGs have been implemented using general purpose processor providing high accuracy and flexibility but those systems consume relatively high power and occupy a large area, restricting their utility in embedded applications. Additionally, in these processors each task gets time on the CPU regardless of its priority and even the most time-critical application can be suspended for some routine maintenance; these two features have a considerable effect on performance and real time response cannot be ensured [9]. These processors have begun to inherit high-performance techniques from their desktop counterparts, such as pipelining, caches, dynamic branch prediction, and multithreading. Unfortunately, even when these techniques offer a good solution in software, their performance cannot be analytically bounded, so when a task will be executed cannot be determined accurately. As a consequence, CPG dedicated hardware implementation, both analog and digital, has received more attention. On one hand, analog circuits have been already proposed, being computational and power efficient, but they usually lack flexibility and they involve large design cycles. Although there is efficient locomotion control based on CPGs, few works have focused on adopting the technology to fully practical embedded implementation with the ability to be scalable or reusable in different robots morphologies [10–12]. Recent developments in embedded controller technology have yielded very sophisticated computing devices in relatively small and easily programmed modules. These technologies are of low cost, power efficient, and adaptive, which might greatly benefit from custom hardware architectures. These architectures can be an alternative to implement robot control schemes that counterbalance the fully analog and digital drawbacks by providing custom efficient hardware attached to embedded processors in a single chip (SoC).

Recently, FPGA (Field Programmable Gate Array) technology has improved in density up to the point that it is feasible to implement large scale bioinspired systems on a single FPGA device. Many interesting bioinspired systems such as locomotion control based on CPGs can be implemented using this technology [13]. FPGAs offer a computational architecture that is well suited for algorithms that require massive parallelism of fine-grained computational units. The inherent parallelism of the logic resources, as well as the availability of hard cores (such as multipliers, large distributed Random Access Memory (RAM) blocks, Digital Signal Processing (DSP), and slices) on the FPGA, allows a considerable computation throughput even at sub-500 MHz clock rates. Although CPGs might not be highly computationally demanding, autonomous robot locomotion needs additional modules to carry out interaction tasks with the environment, and, in general, these tasks are complex to be achieved in embedded general purpose processors by themselves. FPGA implementation can provide flexibility and lower latency and real time responsiveness compared to software-based embedded systems.

2. Materials and Methods

2.1. The Spiking Neuron Model. In its simplest form, the evolution of the membrane potential of the integrate-and-fire spiking neuron model is described by the following equations (see [14, 15] for more details about the derivation):

$$V[k] = \gamma V[k-1](1 - Z[k-1]) + \sum_{j=1}^{N} W_j Z_j[k-1] + I^{\text{ext}}[k-1], \quad (1)$$

$$Z = \begin{cases} 1 & \text{if } V \geq \theta \ (\text{firing}) \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where $V[k]$ represents the membrane potential of a neuron at a discrete time k. $\gamma \in [0, 1[$ defines the leak rate. The firing state is given by the term Z in (2). N is the number of presynaptic neurons. W is the matrix of synaptic weights. Finally, I^{ext} represents an external stimulus.

When $V[k]$ reaches a given threshold θ, then a spike occurs in $Z[k]$ and the neuron is reset by the term $(1 - Z_i[k])$ in (1).

2.2. The Locomotion System. The locomotion system is a spiking neural network (SNN) that acts as a Central Pattern Generator (CPG) [2]. That is, the SNN is able to generate different periodic patterns (locomotion gaits), such as those observed for interleg coordination in free-walking adult stick insects and shown in Figure 1 [8]. Such patterns can be represented as spike trains of N neurons (8 for both robots, quadruped and hexapod), which are estimated from the following equation:

$$V_i[k] = \gamma V_i[k-1]\left(1 - Z_i[k-1]\right) + \sum_{j=1}^{N} W_{ij} Z_j[k-1] + I_i^{\text{ext}}[k-1], \quad (3)$$

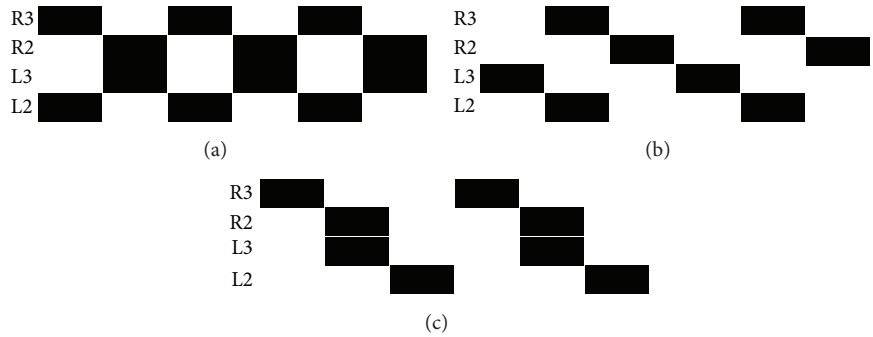


Figure 1: Schematic drawing of different stereotypic quadrupedal walking patterns. In trot, two diagonal legs swing in synchrony (a). In walk, synchronous swing of a diagonal pair of legs is followed by two single leg swing phases (b, c). Black bars indicate leg swing and R and L correspond to right and left sides, respectively [8].

and the spiking activity (spike train) of the whole network is defined by Z as indicated in (2).

The synaptic weights (W) are represented as $N \times N$ matrix, which defines a specific topology for the spiking neural network and it is determined by the methodology described in Section 2.3. Once these parameters have been estimated they are used in (3) to generate the desired locomotion gait. For a practical reason, the value of γ (leakage parameter) has been set to 0.5; this is due to the fact that such value is a power of two (2^{-1}), which is highly suitable for hardware implementation (*binary* operations). The CPGs can produce rhythmic signals without afferent sensory information, and for this reason the SNN does not require exogenous inputs; that is, I^{ext} has been set to zero. Finally, the initial conditions for Z and V correspond to the values at the time $k - 1$ of the desired locomotion gait.

In this research, the locomotion patterns shown in Figure 1 have been modified in order to match with the robot structure. To be more specific, the quadruped and hexapod robots have 12 and 18 Degrees of Freedom (DOFs), respectively, with 3 servomotors (DOFs) per leg, which correspond to femur, tibia, and coxa. However, for locomotion we only need to control 2 of them, femur and coxa (see Figure 5). The servomotor for the tibia only needs to be energised but not controlled. This is due to the fact that the tibia is the weight-bearing part of the robot. Thus, the improved locomotion patterns are shown in Figure 2.

2.3. Parameter Estimation. The parameter estimation of Central Pattern Generators (CPGs) is generally a difficult task; on it depends the spiking neural network's (SNN) capability to periodically replicate a set of rhythmic signals [2]. In this work, the parameter estimation (synaptic weights and connections) of SNNs follows a divide-and-conquer workflow based on an evolutionary approach. Different evolutionary approaches to deal with design and tuning parameters of Artificial Neural Networks have been proposed, for example, weight tuning, topology definition, learning rule optimization, and combination of them. Besides, these evolutionary approaches can search directly or indirectly over the search space according to the representation of candidate solutions (see [16] for a detailed review). Particularly, to tackle the parameter estimation in CPG-based systems driven by evolutionary algorithms, most works use evolutionary algorithms to modulate synaptic parameters of prefixed network topologies. In [17], a Genetic Algorithm for tunning the parameters of CPG designed for the locomotion of both terrestrial and aquatic gaits of a virtual salamander was successfully implemented. However, the most related work to this research is presented in [18], where neural networks are developed and designed by means of Genetic Programming (GP) to make a virtual fish swim, and this work reports the feasibility of using GP to develop CPGs; however, those designs were not implemented on a real robot.

The parameter estimation method deals with an optimization problem, where the search space is formed by all weighted connectivity configurations for a graph with N nodes; hence, the matrix W in (3) defines both the topology and synaptic weights of a SNN; it can be considered as the adjacency transpose matrix of a weighted directed graph. The parameter estimation method divides the design of a SNN into individual connectivity designs for each spiking neuron, instead of designing and training a SNN as a whole. Each connectivity design of a neuron is carried out by an evolutionary approach that represents solutions indirectly. The purpose of the evolutionary method is to find out a set of weighted connections for a neuron to periodically replicate a target rhythmic pattern according to a desired gait. The connectivity configuration of each neuron is represented by words, and (4) shows the syntax for connectivity words, where the first part (before the sign ":") indicates the number of presynaptic connections of the current neuron and the second part (after the sign ":") indicates the indexes and synaptic weights of each presynaptic neuron:

$$\underbrace{n}_{\text{synaptic connections}} : \overbrace{\text{id}_{1\text{st}}, \text{weight}_{1\text{st}}}^{\text{1st configured synapse}} \; | \cdots | \; \overbrace{\text{id}_{n\text{th}}, \text{weight}_{n\text{th}}}^{n\text{th configured synapse}} . \quad (4)$$

The words in (4) besides syntactic correctness require ensuring semantic criteria related to the matrix W in (3) such as the following: the number of presynaptic neurons must be bounded $1 \leq n \leq N$ and the indexes of the presynaptic neurons must be different ($\text{id}_1 \neq \text{id}_2 \neq \cdots \neq \text{id}_{n\text{th}}$) to avoid multiples ties or loops from the same presynaptic neuron to

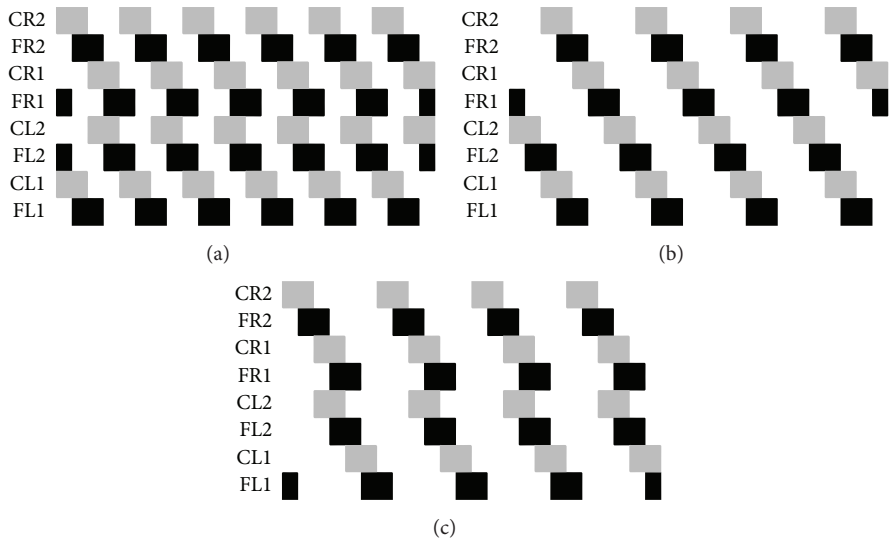


Figure 2: Modified locomotion patterns for the legged robots.

the current neuron being designed (or multiple values into a single cell from the matrix W).

To obtain solutions that are syntactically and semantically correct such as in (4), the configuration for each neuron is carried out by Christiansen Grammar Evolution (CGE) [19] framework and requires the next three components:

(i) A Christiansen Grammar (CG) which reflects the syntactic and semantic requirements for the language of neuron connectivities. Several CG can be designed to cover these requirements, for example, a CG that removes neuron indexes when they were used (see the appendix).

(ii) A fitness function to set the quality of a candidate solution once it has been mapped from its genotypical form (string of numbers) to its phenotypical form (a spiking neuron connected to its presynaptic neurons). An important criterion to achieve that a specific neuron replicates an input rhythmic signal is the design of a fitness function to explore the search space of weighted connectivity configurations. In functional approximation, an alternate and explicit mathematical expression is constructed for the objective function [20], which in this case is unknown. In this work, the SPIKE-distance is used as basis of functional approximation for the fitness function. The SPIKE-distance is a parameter-free and timescale-adaptive measure for estimating the degree of synchrony between spike trains. In general, the distance is defined as a temporal average of the spike trains' dissimilarity profiles (see [21] for detailed definition). Here, the bivariate SPIKE-distance is used to measure the similarity between a target spike train (rhythmic signal) and the spike train generated after simulation by the phenotypical form of the candidate solution.

(iii) A search engine (metaheuristic algorithm) to drive the search of good solutions based on their quality. Here, a continuous Univariate Marginal Distribution Algorithm (UMDA$_c^G$) with elitism is used to evolve the connectivity designs as that is an easy-to-implement evolutionary algorithm (see [22] for implementation details).

The whole design process works as follows: a gait represented as a set of spike trains (rhythmic signals) is given as input. Next, each spike train (rhythmic signal) is treated individually for being designed, and all the spike trains except the targeted one are set as available as activity of feasible presynaptic neurons and the initial state of the current designed neuron is set according to the input gait, if at time $k = 0$ the neuron should fire or should not. Once the required configurations are done the CGE framework evolves candidate solutions for connectivity until an expected error is achieved (this depends on the fitness function being used); after the gait's replication is achieved, the current neuron connectivity design is stored and the process continues until achieving the correct replication of N spike trains from the input gait. Finally, all N individual neuron connectivity designs are integrated into a SNN (see Algorithm 1).

In Figure 3, a graphic workflow of the design method is presented, an input is decomposed into individual spike trains, and each neuron is configured to replicate a specific spike train. Later, all the individual designs are integrated into a SNN; the output is the CPG design, which is illustrated as a directed graph (weights have been omitted for clarity in the graphic), where each neuron configuration in the design part is associated with a color and its connectivity pattern can be visualised in the final graph (the final graph shows that the design does not use all available ties and loops; only the non-gray colored and continuous ones define the CPG topology).

2.4. Hardware. To validate the different configurations of the CPGs we have performed hardware implementation on dedicated hardware, such as Arduino (Microcontroller)

Require: $G = \{S_1, \ldots, S_N\}$
Ensure: CPG design
(1) **for all** $i = 1$ **to** N **do**
(2) Configure the available signals for the ith spiking neuron ($\{G - S_i\}$).
(3) Set the initial state of the ith spiking neuron.
(4) **repeat**
(5) Evolve both the connectivity and synaptic weights for the ith spiking neuron by means of CGE.
(6) **until** Best Solution's Fitness != Expected Error
(7) Store the CGE's Best Solution for the ith spiking neuron's configuration.
(8) **end for**
(9) Integrate all configurations into a SNN.

Algorithm 1: CPG design methodology.

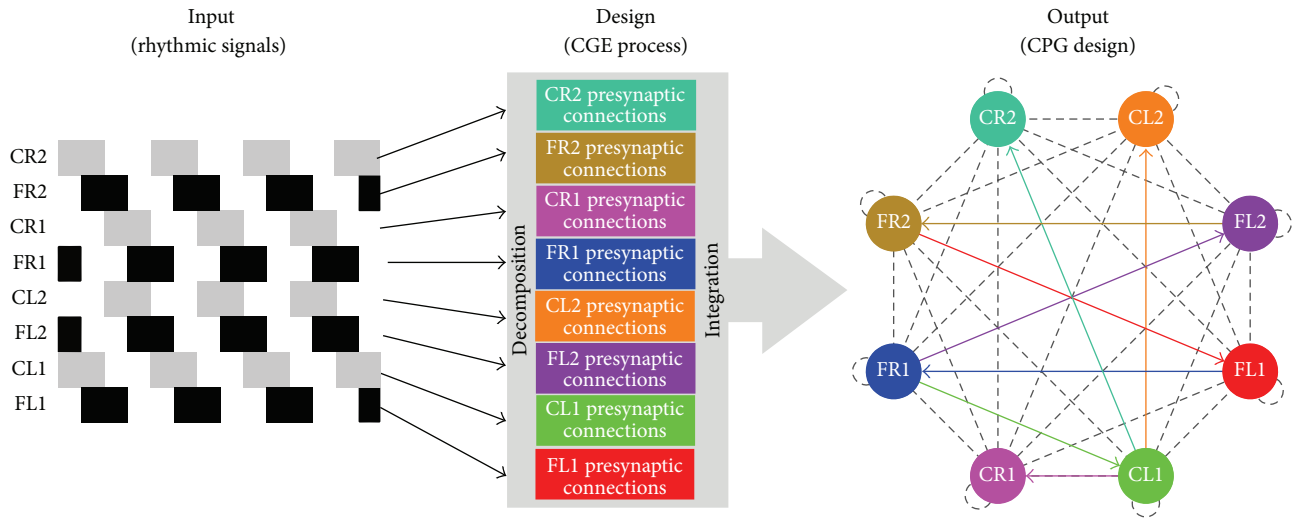


Figure 3: Schematic diagram of the CGE-based methodology for designing CPGs.

and FPGA board (from OpalKelly) for high and low level implementation, respectively.

The Arduino board used in this work is the BotBoarduino (Figure 4(a)), which is based on Atom Microcontrollers for Lynxmotion robots. It has an onboard speaker, three buttons and LEDs, a Sony PS2 controller port, a reset button, logic and servo power inputs, an I/O bus with 20 pins and power and ground, and a 5 vdc 1.5 amp regulator. Also, up to 18 servos can be plugged in directly.

In this work, we also considered the implementation on FPGA board in order to have better hardware conditions, such as more resources to implement complex designs, a hardware design language (HDL), low power consumption, reconfigurability, hardware parallelism, and very high processing speed. Specifically, in this work we use Spartan 6 XEM6310-LX45 board (Figure 4(b)) from the OpalKelly family (https://www.opalkelly.com/). This specific board has two more advantages: on one hand the dimensions, only 75 mm × 50 mm (highly suitable for our robots), and on the other hand a graphical interface for friendly interaction between the PC and the FPGA.

In both cases, a SSC-32 servo controller (Figure 4(c)) is used to handle servomotors in the robot. This is a servo controller with 32 channels of 1uS resolution servo control and a bidirectional communication.

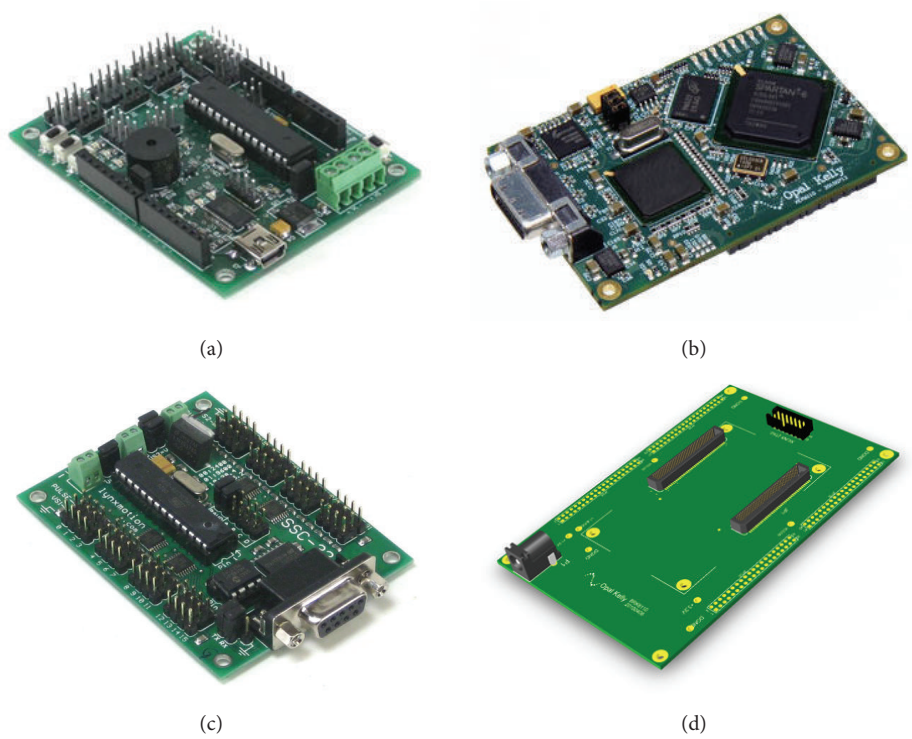
To connect and control the servos through the FPGA, we also use a breakout board (BRK6110), which allows us an easy connection to high-density connectors on the XEM6110-LX45 by routing all signals to four 40-pin 2-mm headers (see Figure 4(d)).

To validate the CPGs, we have used real quadruped and hexapod robots, such as those shown in Figure 5.

3. Results

Here, we present test results for the performance of the system. We first estimated the synaptic weights for three different gaits (walking, jogging, and running) of the SNNs by using Christiansen Grammar Evolution. For this, we performed three strategies based on three different SPIKE-distance-based fitness functions as follows:

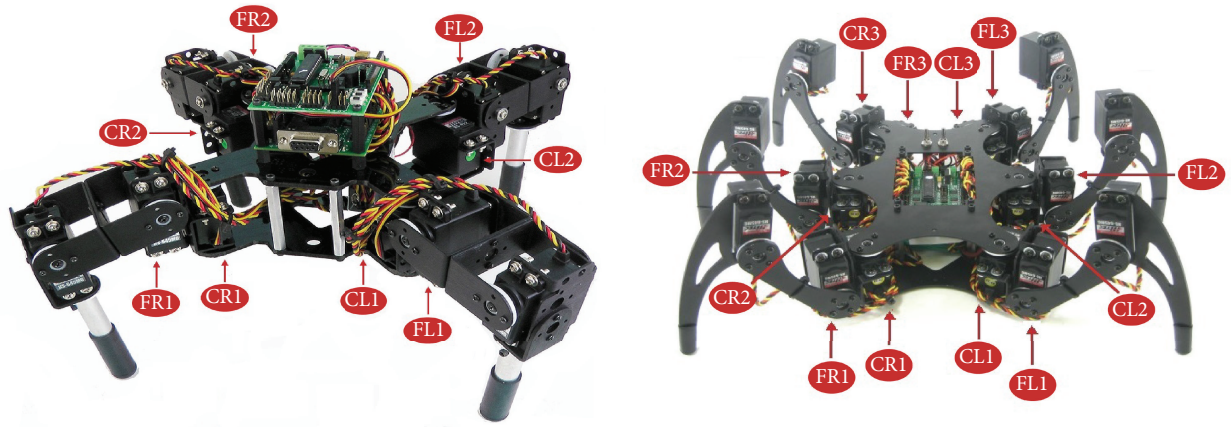
(i) The Christiansen Grammar Evolution runs with a SPIKE-distance-based fitness function which has no restrictions on the number of synaptic connections.



(a) (b)

(c) (d)

Figure 4: Hardware used in this research: (a) BotBoarduino, (b) FPGA, (c) servo controller SSC32, and (d) breakout board BRK6110.



(a) (b)

Figure 5: Neuronal configurations in the legged robots. (a) Quadruped and (b) hexapod robots. C and F indicate coxa and femur, respectively. L and R correspond to the side where the neurons are located in the robot (images from Lynxmotion website).

(ii) The Christiansen Grammar Evolution runs with a SPIKE-distance-based fitness function with restrictions on the number of synaptic connections; that is, we expect that only one presynaptic neuron can stimulate the postsynaptic neuron to reproduce its input signal.

(iii) The Christiansen Grammar Evolution has a SPIKE-distance-based fitness configured to reproduce the three different locomotion patterns (gaits) with the same spiking neural network topology.

From these strategies, we obtained the different configurations for the spiking neural networks, which generate the locomotion patterns for the legged robots.

Equations (5), (6), and (7) correspond to the connectivity matrices for walking, jogging, and running gaits, respectively, generated by the Christian Grammar Evolution with no restrictions on the number of synaptic connections among the neurons. Similarly, (8), (9), and (10) correspond to the connectivity matrices for the same gaits but with restrictions on the number of synaptic connections. Finally, (11) corresponds to the connectivity matrix which can reproduce any

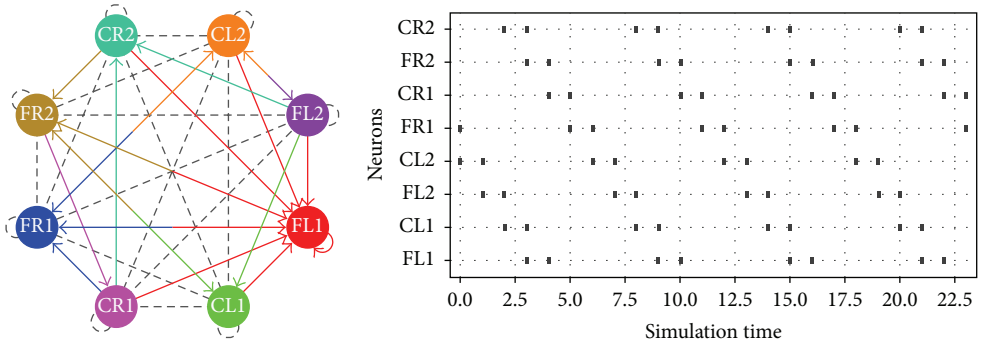


Figure 6: (a) Network topology and (b) raster plot for the walking gait.

of the three gaits; this is given when the third strategy is used with Christiansen Grammar Evolution.

Besides, in Figures 6, 7, and 8, the schematic on the left side corresponds to the spiking neural networks topologies for walking, jogging, and running gaits, respectively. On the right side the raster plots (spiking activity) for the three locomotion patterns are shown:

$$w_{w_1} = \begin{vmatrix} -5 & 4 & 2 & -9 & 2 & -7 & 3 & 2 \\ 0 & 0 & 5 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -7 & 0 & 9 & 0 & 0 & 0 \\ -3 & 0 & 0 & -3 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ -4 & 7 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 4 & 0 & 0 & -2 & 0 & 0 \end{vmatrix}, \quad (5)$$

$$w_{j_1} = \begin{vmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & -3 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -3 & 4 & 0 \\ 0 & 0 & 0 & 8 & 0 & -1 & 8 & -6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & -5 & 0 & 0 & 0 & 0 & 4 \\ 6 & 7 & -2 & 5 & -7 & 0 & -6 & 0 \end{vmatrix}, \quad (6)$$

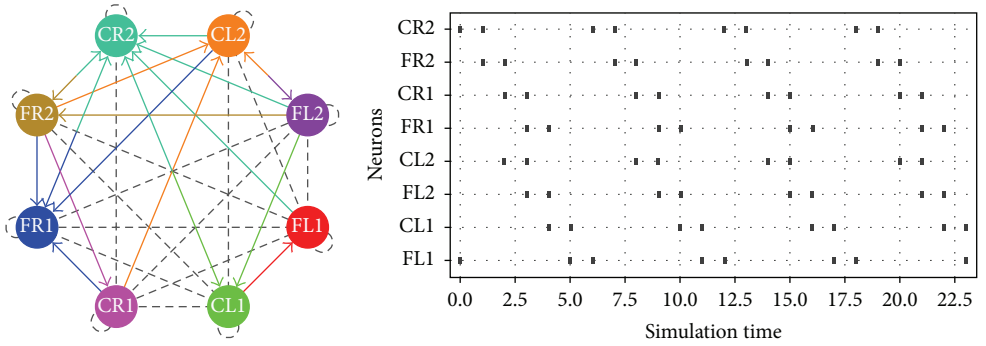
$$w_{r_1} = \begin{vmatrix} -7 & 9 & 4 & -1 & 0 & -6 & -3 & 7 \\ -5 & -8 & 9 & 5 & 0 & 0 & 0 & 3 \\ 0 & -7 & 0 & 4 & 0 & 8 & 0 & 0 \\ 2 & 6 & -8 & -2 & -7 & -3 & 6 & 5 \\ -1 & -7 & 6 & 5 & -4 & 7 & -4 & 4 \\ 8 & 0 & -2 & 0 & -4 & 0 & 7 & 0 \\ 6 & 7 & -1 & 0 & 7 & -5 & -7 & -4 \\ -7 & -9 & 9 & 0 & 3 & 0 & -4 & 1 \end{vmatrix}, \quad (7)$$

$$w_{w_2} = \begin{vmatrix} 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}, \quad (8)$$

$$w_{j_2} = \begin{vmatrix} 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}, \quad (9)$$

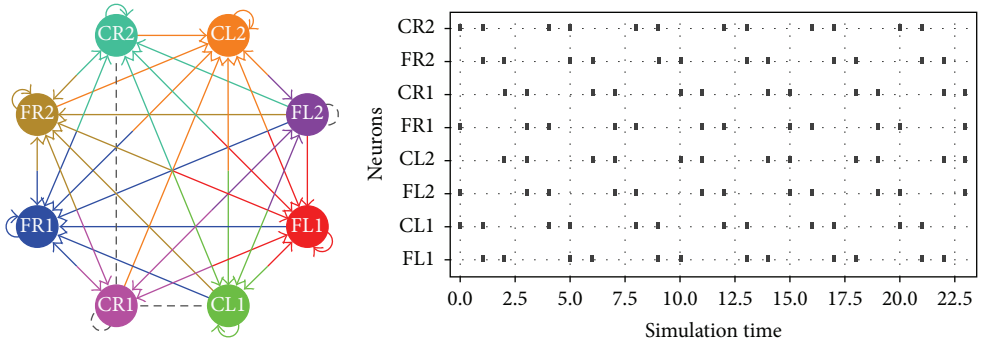
$$w_{r_2} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}. \quad (10)$$

In Figure 11, we show the network topologies when the parameter estimation presents restrictions on the number of synaptic connections. As we can observe, the connectivity map is clearer and easier to be implemented in hardware. The raster plots are exactly the same as those shown in Figures 6(b), 7(b), and 8(b).



(a) (b)

Figure 7: (a) Network topology and (b) raster plot for the jogging gait.



(a) (b)

Figure 8: (a) Network topology and (b) raster plot for the running gait.

In Figure 12, we present the most interesting implementation, because such network is able to generate the three locomotion patterns presented in this work.

Finally, in Figure 13, we show real time simulations during the walking, jogging, and running gaits of the quadruped and hexapod robots through the use of a digital oscilloscope; in such figure, x-axis and y-axis represent neuron activity through time and neuron labels, respectively. For the hexapod robot, we can appreciate that two signals are death, because the robot was amputated; for reasons of robot's stability, we have simulated that middle legs were amputated instead of the front legs as in the experimentation reported in [8]. A sequence of the movement of the quadruped and the hexapod robots is shown in Figures 9 and 10, respectively (the sequence in both figures goes from left to right and from top to bottom):

$$w_{aio} = \begin{vmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 \\ -1 & -4 & 0 & 7 & 0 & 0 & 0 & 0 \\ -3 & 3 & -9 & 0 & 7 & -2 & 6 & -1 \\ 0 & 0 & 0 & 0 & -3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 & -2 \\ 0 & 0 & 0 & 0 & 0 & -8 & 0 & 8 \\ 5 & 0 & 7 & 0 & -6 & 0 & -5 & 0 \end{vmatrix}. \quad (11)$$

4. Conclusions

A biologically inspired embedded system for quadrupedal robot locomotion has been presented. The design methodology includes a solid mathematical background, biological validation, numerical simulations, and hardware implementation. The theoretical framework includes spiking neurons to reproduce locomotion patterns and a grammar evolution approach to estimate the parameters of such neurons. In this work, we show how the simplest spiking neuron model is able to reproduce periodic patterns such as those observed in living entities. At the same time, we applied the Christiansen Grammar Evolution to estimate the weights and synaptic connections of the spiking neural network to achieve an exact reproduction of the desired locomotion patterns. Also, we have improved the parameter estimation in spiking neural networks by incorporating three different fitness functions and using the SPIKE-distance to compare the effectiveness of each of them.

Numerical simulations have demonstrated the effectiveness of the whole system; however, the major achievement is the hardware implementation of this system on real legged robots. Implementation in high and low level programming languages has been performed in order to control the robots by using Arduino and FPGA boards, respectively.

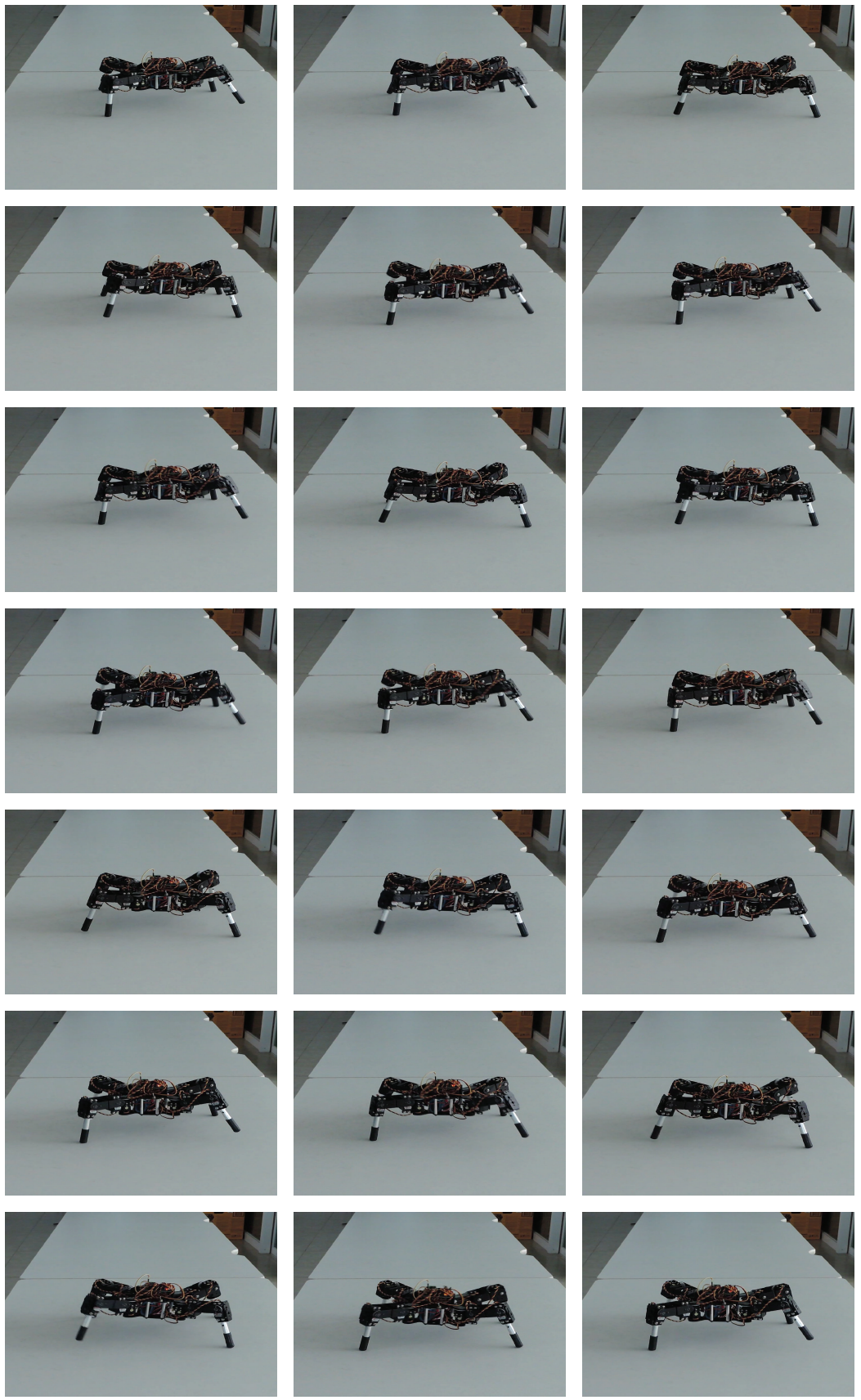


Figure 9: Real time simulation on a quadruped robot.

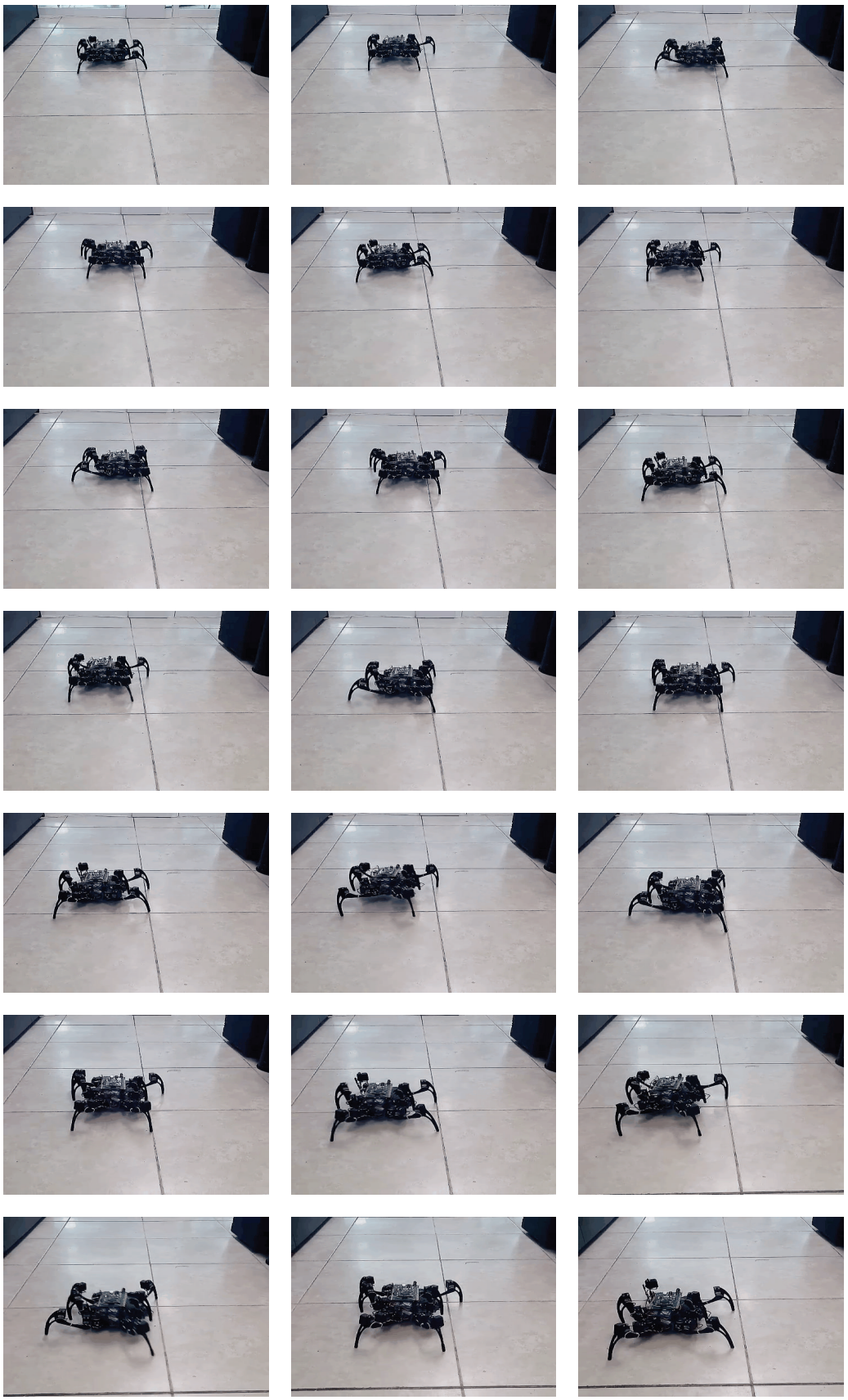


Figure 10: Real time simulation on a hexapod robot with middle legs amputated.

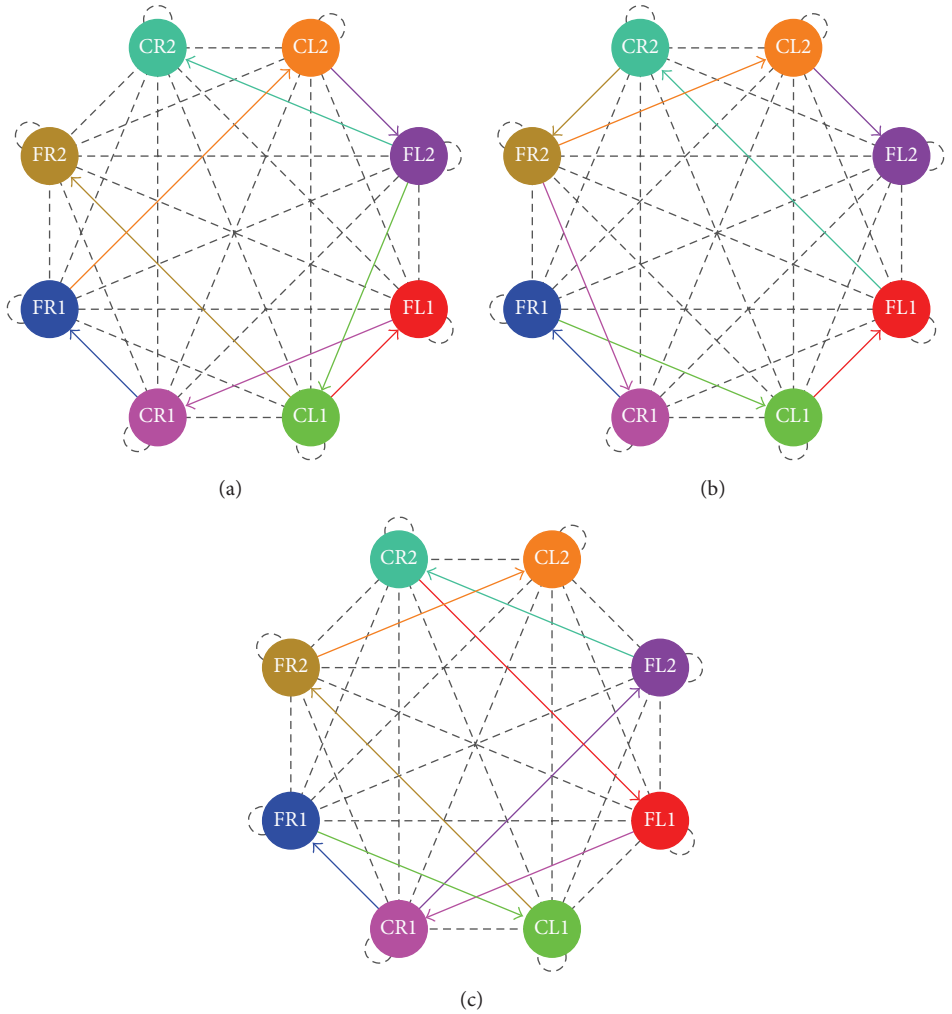


Figure 11: Network topologies with restrictions on the number of synaptic connections for (a) walking, (b) jogging, and (c) running gaits.

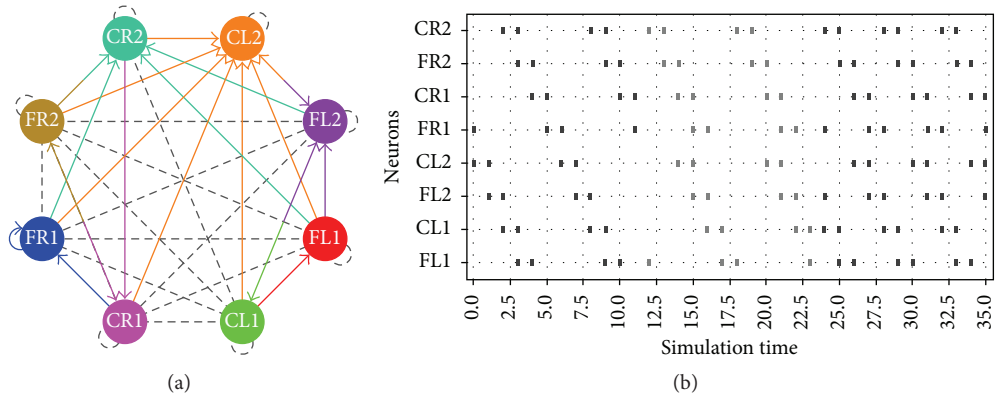


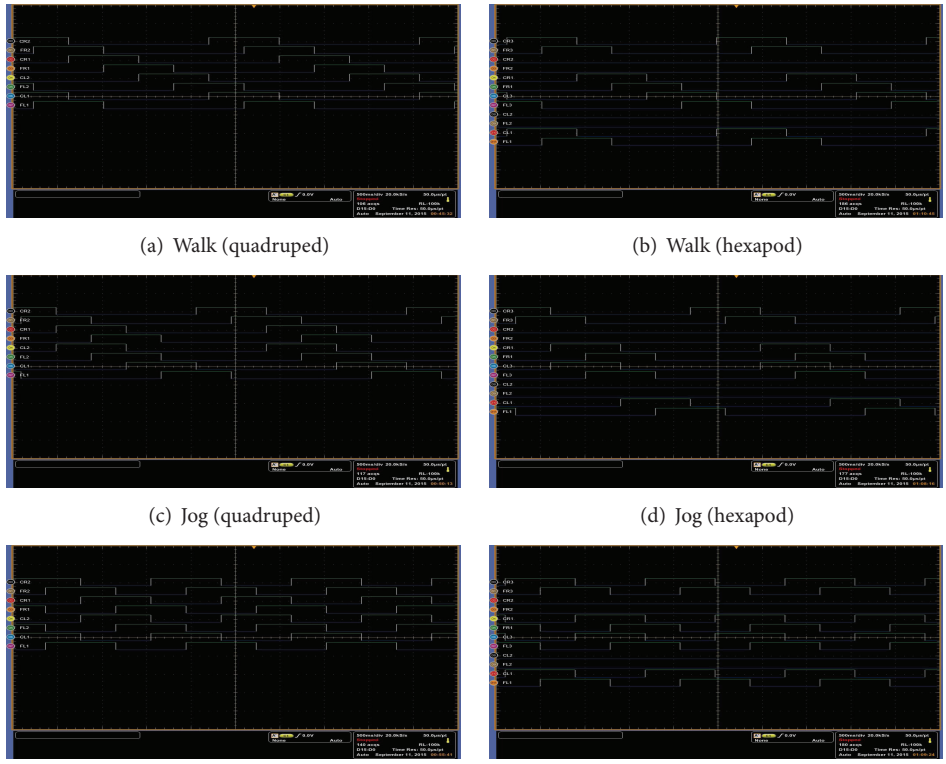
Figure 12: (a) All-in-one topology network. (b) The three locomotion gaits are generated by the same network.

Real time numerical experiments on two-legged robots (quadruped and hexapod) have been shown in Section 3. This methodology can also be applied to any kind of legged robot. Next step in this research is the use of sensory information for autonomous navigation.

Appendix

Christiansen Grammar

A Christiansen Grammar (CG) is a set $CG = \{\Sigma_N, \Sigma_T, S, P\}$, where Σ_N is the set of nonterminals, Σ_T is the set of terminals,



(a) Walk (quadruped) (b) Walk (hexapod)

(c) Jog (quadruped) (d) Jog (hexapod)

(e) Run (quadruped) (f) Run (hexapod)

Figure 13: Oscilloscope screenshots for the three locomotion gaits (real time monitoring). On the left side, we show the real time simulation for the quadruped robot, and on the right side the real time simulation for the amputated hexapod robot.

S is the initial start symbol ($S \in \Sigma_N$), and P is the set of productions (see [19] for more details). Next the CG is shown which allows deriving words that accomplish the restrictions mentioned in Section 2.3 by creating a production which contains a finite number of presynaptic connections ($1 \leq n \leq N$), besides preventing repetitive connections from a presynaptic neuron by eliminating indexes of presynaptic neurons already connected ($\mathrm{id}_1 \neq \mathrm{id}_2 \neq \cdots \neq \mathrm{id}_{n\mathrm{th}}$). Next, the herein proposed CG for defining presynaptic connections of a neuron is introduced:

$$
\begin{aligned}
&\mathrm{CG} = \{\\
&\Sigma_N = \{\langle \mathit{neuronSynapses}\rangle(\downarrow g_i), \langle \mathit{connections}\rangle(\downarrow g_i, \uparrow g_o),\\
&\langle \mathit{neuronIdList}\rangle(\downarrow g_i, \uparrow n), \langle \mathit{synapses}\rangle(\downarrow g_i), \langle \mathit{synapse}\rangle(\downarrow g_i, \uparrow g_o),\\
&\langle \mathit{weight}\rangle(\downarrow g_i), \langle \mathit{sign}\rangle(\downarrow g_i), \langle \mathit{digit}\rangle(g_i)\},\\
&\Sigma_T = \{1, 2, \ldots, 8, 9, +, -, :, |\},\\
&S = \langle \mathit{neuronSynapses}\rangle(\downarrow g_i),\\
&P = \{\langle \mathit{neuronSynapses}\rangle(\downarrow g_i) \vDash \langle \mathit{connections}\rangle(\downarrow g_i, \uparrow g_o):\\
&\langle \mathit{synapses}\rangle(\downarrow g_o)\{\},\\
&\langle \mathit{connections}\rangle(\downarrow g_i, \uparrow g_o) \vDash \langle \mathit{neuronIdList}\rangle(\downarrow g_i, \uparrow n)\{\\
&\uparrow g_o = \downarrow g_i \cup \langle \mathrm{synapses}\rangle(\downarrow g_i) \vDash \langle \mathrm{synapse}\rangle(\downarrow g_i, \uparrow g_{o_1})| \cdots\\
&|\langle \mathrm{synapse}\rangle(\downarrow g_{o_{\uparrow n-1}}, \uparrow g_{o_{\uparrow n}})\{\}\},\\
&\langle \mathit{neuronIdList}\rangle(\downarrow g_i, \uparrow 1) \vDash 1\{\},\\
&\vdots\\
&\langle \mathit{neuronIdList}\rangle(\downarrow g_i, \uparrow N) \vDash N\{\},\\
&\langle \mathit{synapse}\rangle(\downarrow g_i, \uparrow g_o) \vDash \langle \mathit{neuronIdList}\rangle(\downarrow g_i, \uparrow n), \langle \mathit{weight}\rangle(\downarrow g_i)\{\\
&\uparrow g_o = \downarrow g_i - \{\langle \mathrm{neuronIdList}\rangle(\downarrow g_i, \uparrow n) \vDash \uparrow n\{\}\}\},\\
&\langle \mathit{weight}\rangle(\downarrow g_i) \vDash \langle \mathit{sign}\rangle(\downarrow g_i)\langle \mathit{digit}\rangle(\downarrow g_i)\{\},\\
&\langle \mathit{sign}\rangle(\downarrow g_i) \vDash +\{\},\\
&\langle \mathit{sign}\rangle(\downarrow g_i) \vDash -\{\},\\
&\langle \mathit{digit}\rangle(\downarrow g_i) \vDash 1\{\},\\
&\vdots\\
&\langle \mathit{digit}\rangle(\downarrow g_i) \vDash 9\{\}\}\\
&\}
\end{aligned}
$$

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research has been partially supported by the CONACYT project "Aplicacion de la Neurociencia Computacional en el Desarrollo de Sistemas Roboticos Biologicamente Inspirados" (no. 269798).

References

[1] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *The Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

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

[3] H. Kimura, I. Shimoyama, and H. Miura, "Dynamics in the dynamicwalk of a quadruped robot," *International Journal of Robotics Research*, vol. 4, no. 2, pp. 187–202, 2003.

[4] P. Arena, L. Fortuna, M. Frasca, and G. Sicurella, "An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 4, pp. 1823–1837, 2004.

[5] L. H. Scott, "Central pattern generator," *Current Biology*, vol. 10, no. 2, pp. 176–177, 2000.

[6] C. M. A. Pinto, "Central pattern generator for legged locomotion: a mathematical approach," in *Proceedings of the Workshop on Robotics and Mathematics*, vol. 16, pp. 1–6, 2007.

[7] A. Fujii, N. Saito, K. Nakahira, A. Ishiguro, and P. Eggenberger, "Generation of an adaptive controller CPG for a quadruped robot with neuromodulation mechanism," in *Proceedings of the International Conference on Intelligent Robots and Systems (IEEE/RSJ '02)*, pp. 2619–2624, IEEE, October 2002.

[8] M. Grabowska, E. Godlewska, J. Schmidt, and S. Daun-Gruhn, "Quadrupedal gaits in hexapod animals—inter-leg coordination in free-walking adult stick insects," *The Journal of Experimental Biology*, vol. 215, no. 24, pp. 4255–4266, 2012.

[9] C. Weems and S. Dropsho, "Real-time computing: implications for general microprocessors," Tech. Rep., University of Massachusetts, 1995.

[10] A. Billard and A. J. Ijspeert, "Biologically inspired neural controllers for motor control in a quadruped robot," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '00)*, pp. 637–641, July 2000.

[11] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 187–202, 2003.

[12] S. Still and M. W. Tilden, "Controller for a four-legged walking machine," in *Neuromorphic Systems Engineering Silicon from Neurobiology*, vol. 10 of *Progress in Neural Processing*, pp. 138–148, World Scientific, Singapore, 1998.

[13] J. Liu and C. Wang, "A survey of neuromorphic engineering—biological nervous systems realized on silicon," in *Proceedings of the IEEE Circuits and Systems International Conference on Testing and Diagnosis (ICTD '09)*, pp. 1–4, IEEE, Chengdu, China, April 2009.

[14] H. Soula, G. Beslon, and O. Mazet, "Spontaneous dynamics of asymmetric random recurrent spiking neural networks," *Neural Computation*, vol. 18, no. 1, pp. 60–79, 2006.

[15] B. Cessac, "A discrete time neural network model with spiking neurons: rigorous results on the spontaneous dynamics," *Journal of Mathematical Biology*, vol. 56, no. 3, pp. 311–345, 2008.

[16] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[17] A. J. Ijspeert, "A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander," *Biological Cybernetics*, vol. 84, no. 5, pp. 331–348, 2001.

[18] A. J. Ijspeert and J. Kodjabachian, "Evolution and development of a central pattern generator for the swimming of a lamprey," *Artificial Life*, vol. 5, no. 3, pp. 247–269, 1999.

[19] A. Ortega, M. de la Cruz, and M. Alfonseca, "Christiansen grammar evolution: grammatical evolution with semantics," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 77–90, 2007.

[20] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.

[21] T. Kreuz, D. Chicharro, C. Houghton, R. G. Andrzejak, and F. Mormann, "Monitoring spike train synchrony," *Journal of Neurophysiology*, vol. 109, no. 5, pp. 1457–1472, 2013.

[22] D. Simon, *Evolutionary Optimization Algorithms*, John Wiley & Sons, 2013.