

# Chapter 5

## Speed Control on a Hexapodal Robot Driven by a CNN-CPG Structure

E. Arena, P. Arena and L. Patané

**Abstract** Locomotion control in legged robots is an interesting research field that can take inspiration from biology to design innovative bio-inspired control systems. Central Pattern Generators (CPGs) are well known neural structures devoted to generate activation signals to allow a coordinated movement in living beings. Looking in particular in the insect world, and taking as a source of inspiration the *Drosophila melanogaster*, a hierarchical architecture mainly based on the paradigm of a Cellular non-linear Network (CNN) has been developed and applied to control locomotion in a fruit fly-inspired simulated hexapod robot. The modeled neural structure is able to show different locomotion gaits depending on the phase locking among the neurons responsible for the motor activities at the level of the leg joints and theoretical considerations about the generated pattern stability are discussed. Moreover the phase synchronization between the leg, altering the locomotion, can be used to modify the speed of the robot that can be controlled to follow a reference speed signal. To find the suitable transitions among patterns of coordinated movements, a reward-based learning process has been considered. Simulation results obtained in a dynamical environment using a *Drosophila*-inspired hexapod robot are here reported analyzing the performance of the system.

### 5.1 Introduction

Gait generation and locomotion control in artificial systems are extremely important to build efficient and highly adaptive walking robots. Since the last decade a huge effort has been paid to discover and model the rules that biological neural systems

---

E. Arena · P. Arena (✉) · L. Patané  
Department of Electrical, Electronic and Computer Science Engineering, University of Catania,  
I-95125 Catania, Italy  
e-mail: parena@dieei.unict.it

L. Patané  
e-mail: lpatane@dieei.unict.it

P. Arena  
National Institute of Biostructures and Biosystems (INBB), Viale delle Medaglie d'Oro 305,  
00136 Rome, Italy

adopt to show efficient strategies for generating and controlling the gait in animals and manage the efficient transition among different patterns of locomotion. The work here presented is in line with the on-going studies on the insect brain architecture [1, 2]. In particular a huge effort has been paid recently to design block-size models for a number of different parts of the fly *Drosophila melanogaster* brain, to try to attain perceptual capabilities and to transfer them to biorobots. In the field of Bio inspired cognitive Robotics, the paradigm of Cellular Nonlinear Networks, the continuous time extension of cellular automata, has been widely exploited for their capabilities of complex spatial temporal pattern formation, both in the steady state regime [3, 4], and through dynamical attractors [5].

Regarding the Neurobiological studies on the fly motor control, while it is already known which are the centers involved in visually guided orientation control behaviors (i.e. Central Complex) [6–8], it is not clear how the high level controller acts at the low level, to finely modulate the neural circuitry responsible for the locomotion pattern generation, steering activities and others. On the other side, behavioral experiments are in line with the idea that the fruit fly mainly adopts the Central Pattern Generator (CPG) scheme to generate and control its locomotion patterns [9–11]. A plausible CPG based neural controller was then designed, able to generate the joint signals and the consequent stepping diagrams for the fruit fly. The designed network was used to control an artificial model of the fruit fly built using a dynamic simulation environment as will be reported in the next sections.

In literature several CPG-based central structures were developed and applied to different robotic platforms [12]. Here the possibility to host signals coming from sensors can improve the robot performance in terms of adaptability to the environment state [13]. The use of dynamical oscillators is also commonly exploited to represent the overall joint activity of a whole neural group and the different topological links among the oscillators give the opportunity to develop a rich variety of robot behaviors [14]. The various locomotion gaits are obtained imposing different phase displacements among the oscillators, which however, have to maintain in time the imposed phase synchronization. Adaptive walking and climbing capabilities were also obtained in real and simulated hexapodal structure referring to CPG realised via CNN architectures [15, 16]. However, only a few works deal with the problem of stability of the obtained gait, which is indeed a crucial aspect to be analyzed. In the proposed work a network of coupled oscillators is used to control the 18 DOFs of a *Drosophila*-like hexapod structure. To create a stable gait generator, a two layer structure is used to uncouple the gait generation mechanisms from the low level actuation of the legs that present different peculiar kinematics structures. To guarantee the stability of the imposed locomotion gaits, the partial contraction theory [17] has been suitably applied. The proof of convergence to every imposed gait thanks to the particular tree structure of the proposed CNN network is guaranteed. The defined CPG is then available to control the locomotion of an hexapod simulated robot with a variety of gaits that can be obtained changing the phase relations between the interconnected neurons dedicated to each leg. In Nature the locomotion pattern is changed in time depending both on the environmental constraints and on the speed imposed by the internal state of the insect. Therefore we proposed to include, as a

higher controller a neural structure similar to a Motor Map. This neural net creates an unsupervised association between the reference speed that, together with the actual speed, is given as input, and the phase value used to synchronize the CPG neurons in order to control the robot speed.

Motor Maps were already used in a number of different complex control issues. In particular, they were used, together with CNNs, to model bio-inspired perceptual capabilities implemented on roving robots [18, 19]. An approach similar to that one presented here already appeared in previous works: there a simplified symmetric structure was considered for the robot and different controlling parameters at the level of CPG were taken into account to fulfill the task [20].

In this work the control parameters are exactly the phases among the legs that can be freely imposed without loosing the phase stability, thanks to powerful theoretical results recently found in this class of non-linear systems.

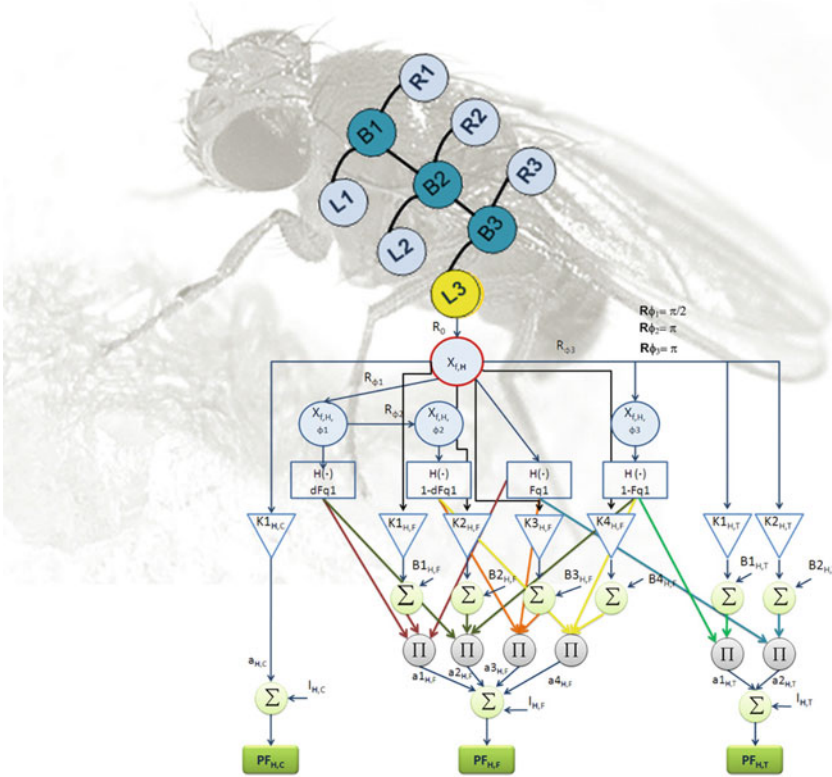
In the next sections the complexity of the problem will be explained to understand why a linear controller like a PID cannot be used to solve the proposed task.

The learning process was carried out in a dynamic simulation where a *Drosophila*-like hexapodal structure was designed. Interesting information on the real stepping diagrams of the system can be extracted from the environment together with the robot position and speed in time, in order to evaluate the performance of the developed neural controller.

## 5.2 The Neural Network for Locomotion Control

The CNN based locomotion controller for our bio-robot follows the traditional guidelines that characterize the Central Pattern Generator paradigm. This is divided into subnetworks. Starting from the lowest level, motor neurons and interneurons are devoted to stimulate the muscle system for each of the limbs of the animal. The neural control of the motion of each limb suitably fits the kinematic constraints and geometric parameters of the limb itself so as to evoke a set of fixed action patterns. The way in which the different limb motions are synchronized to achieve an organized locomotion activity for the animal is managed by a higher level net of Command Neurons. These generates the suitable phase displacement for the implementation of a number of different locomotion patterns, which vary according to the environmental as well as to the internal state of the animal. The overall scheme of the neural controller is reported Fig. 5.1. Where the top layer represents the command system, whereas the bottom layer accounts for the local motorneuron systems. Among the different neuron models nowadays available, the authors already had introduced a neuron model that suitably matched the CNN basic cell, including the nonlinearity [11]. Its equations are reported below:

$$\begin{cases} \dot{x}_{1,i} = -x_{1,i} + (1 + \mu + \varepsilon)y_{1,i} - s_1 y_{2,i} + i_1 \\ \dot{x}_{2,i} = -x_{2,i} + s_2 y_{1,i} + (1 + \mu - \varepsilon)y_{2,i} + i_2 \end{cases} \quad (5.1)$$



**Fig. 5.1** Neural network scheme: the *top* layer generates a stable locomotion pattern, whereas the *bottom* layer is constituted by additional sub-networks generating the needed signals for the leg joints actuation

Here the authors substituted the original Piece wise linear output nonlinearity, typical of standard CNNs, with its smooth approximation  $y_i = \tanh(x_i)$ , uniquely for simplicity in using mathematical tools for proving stability results.

By using the following parameters for each cell:  $\mu = 0.23, \varepsilon = 0, s_1 = s_2 = 1, i_1 = i_2 = 0$  the cell dynamics is able to show a stable *limit cycle* behavior [21]. In this case, the  $\mu$  value was chosen so as to make the ratio between the slow and the fast part of the dynamics of the limit cycle next to one, to approximate a harmonic oscillator; nevertheless, other values can be used to make the system dynamics to elicit a spiking activity.

Once defined the cell dynamics, the command network is built by locally connecting the cells using bidirectional diffusion connections. In particular, the diffusion effect implements a suitable phase shift among the command neurons, which will then become drivers for the lower level motor nets controlling each leg. To this purpose, the cloning templates for the command net can be directly defined through

rotational matrices  $R(\phi)$ , locally linking the command neurons. The whole dynamics configures as a two layer RD-CNN:

$$\dot{x} = f(x) - k \cdot L \cdot x \quad (5.2)$$

where  $x$  is the state variables vector  $(x_1, \dots, x_{2N})^T$ ,  $N$  is the number of cells;  $f(x) = [f(x_1), \dots, f(x_{2N})]^T$  is the dynamics of the whole uncoupled system;  $L$  is the laplacian diffusion matrix,  $k$  is the diffusion coefficient, standing for a coupling gain.

This equation, written in terms of a standard RD-CNN reads:

$$\dot{x}_i = -x_i + \sum_{Cell(j) \in N_r(i)} [A_{i;j}y_j + B_{i;j}u_j + C_{i;j}x_j] \quad (5.3)$$

$$1 \leq i, j \leq N$$

Here, being the system autonomous,  $B = 0$ . On the other hand, the laplacian operator modulates directly the influences among the state variables; therefore, the paradigm of the state controlled CNN, introduced in [22] is here used. Being the CNN cell a second order system it results:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}; \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}; \quad (5.4)$$

with

$$\begin{aligned} A_{11} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}; & A_{12} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ A_{21} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}; & A_{22} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}. \end{aligned} \quad (5.5)$$

$$\begin{aligned} C_{11} &= \begin{pmatrix} 1 & \cos(\phi_{i-1;i}) & 0 \\ -\cos(-\phi_{i-1;i}) & d & -\cos(\phi_{i;i+1}) \\ 0 & -\cos(-\phi_{i;i+1}) & 1 \end{pmatrix}; \\ C_{12} &= \begin{pmatrix} 0 & -\sin(\phi_{i-1;i}) & 0 \\ -\sin(-\phi_{i-1;i}) & 0 & \sin(\phi_{i;i+1}) \\ 0 & \sin(-\phi_{i;i+1}) & 0 \end{pmatrix}; \\ C_{21} &= \begin{pmatrix} 0 & \sin(\phi_{i-1;i}) & 0 \\ \sin(-\phi_{i-1;i}) & 0 & \sin(\phi_{i;i+1}) \\ 0 & \sin(-\phi_{i;i+1}) & 0 \end{pmatrix}; \end{aligned}$$

$$C_{22} = \begin{pmatrix} 1 & \cos(\phi_{i-1;i}) & 0 \\ -\cos(-\phi_{i-1;i}) & d & -\cos(\phi_{i;i+1}) \\ 0 & -\cos(-\phi_{i;i+1}) & 1 \end{pmatrix}. \quad (5.6)$$

The parameter  $d$  is equal to the number of cells directly connected to the considered one; this corresponds to the un-weighted degree of the underlying graph. Moreover zero boundary conditions were considered. The template parameters can be easily derived considering that, in view of the bidirectional connections, among a given cell and the neighbors, there exists a precise phase displacement  $\phi$  which is imposed using the classical rotation matrix in  $\mathbf{R}^2$ :

$$R(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix}; \quad (5.7)$$

More details can be found in [1].

Being the connection matrix  $L$  defined as a function of the imposed phase shift among the oscillators,  $L$  imposes a particular locomotion pattern through the associated *Flow Invariant Subspace*  $M$  [17], which was proven to be a global exponential attractor for the network dynamics. In fact the particular topology of the RD-CNN, used as the command neuron net, can be seen as a dynamic undirected diffusive tree-graph consisting of 9 neurons (see Fig. 5.1). For this particular family of configurations, an important result on the global asymptotic stability was derived [23]: any desired phase shift among the cells can be obtained if the following constraint is imposed for the diffusion coefficient  $k$ :

$$k \cdot \lambda_1 > \sup_{x_i, t} \lambda_{max} \left( \frac{\partial f}{\partial x} (x_i, t) \right) \quad (5.8)$$

where  $\lambda_1$  is the algebraic connectivity of the graph associated to the network [17]. This guarantees asymptotic phase stability to the network, i.e. any desired phase among the command neurons (which will reflect into a phase shift among the robot legs) can be obtained. Once the topology is fixed (in terms of cell structure and network tree topology), all the parameters in Eq. 5.8 are known: the suitable  $k$  value can be therefore selected so as to make the network converge exponentially to any arbitrary flow invariant subspace  $M$ , defined through the phase displacements [24, 25].

### 5.2.1 Leg Motor Neuron Network

Neural signals, consisting in oscillating potentials from the command neuron net, reach the lower level neural structures innervating each of the limbs. These neurons have to elicit fixed action patterns synchronized with the wave of neural activity imposed by the command neurons. It is known that in many insects, including adult

*Drosophila*, the six legs move in a highly coordinated way, thanks to a network of axons coming from the part of the central nervous system located in the thoracic ganglia and synapsing onto specific muscle set [26]. The motor neuron network that we are presenting here consists of a series of neurons which are enslaved by the command neurons and send combined signals to the leg actuators. These signals are peculiarly shaped so as to adapt to the particular leg kinematic structure. The particular motor control network designed for each leg can be still considered as a CNN, since all neurons are characterized by local connectivity and retain the same structure as in Eq. 5.1. One main difference over the command neuron net is that connections among the motor neurons are mono directional. This gives the possibility to act in a top down fashion and prevents disturbances acting at the bottom layer to reach and affect the overall command neuron dynamics. This functional polarity is common of synapses and frequently met in locomotion control of multi legged systems, endowed with both chemical and electrical rectifying synapses [27]. This is really useful for our purposes, since a leg could be even temporarily disconnected from the command system without affecting the high level organized dynamics. This can be necessary to perform particular steering maneuvers (like turning on the spot) or special strategies for looking for a suitable foot bold position. An example of the neural motor system designed for the front leg of our robot prototype is depicted in the bottom part of Fig. 5.1. The state variables of the motor neurons are also post-processed and combined through gating functions, gains, offsets and multipliers to provide the appropriate *Primitive functions* controlling the coxa, femur and tibia joints for each leg. For the case of the rear leg, all neuron oscillators have the same frequency. For an accurate implementation of the motions for the middle and front leg, the presence of cells oscillating at a frequency resulting the double with respect to the one adopted for all the others is needed. In this case a specific control strategy based on impulsive synchronization has been implemented [28]. The network designed, in spite of its apparent complexity, can allow a high degree of adaptability by modulating a small set of parameters.

### 5.3 Reward-Based Learning for Speed Control

The RD-CNN structure presented above does not show any aspect related to learning or adaptation. However a highly degree of adaptability is required for a flexible locomotion control. In this paper we refer on how to introduce a suitable strategy to modulate the robot velocity. The insect brain computational model recently designed [1], hypothesizes the presence of two main blocks: the *Decision layer* and the *Motor layer*, which includes the *Description of behaviors*. The former, according to specific drives coming from the internal state of the animal or from specific external inputs, selects the particular behavior to be taken, whereas the latter is in charge for the description of the behavior to be implemented in terms of the consequent motor organization at the level of the limbs. Whereas some of the most basic behaviors are inherited, some others have to be learned to face with novel circumstances. In

our specific case of learning speed control, we can assume that the drives impose a specific speed reference value and this has to be translated into a particular locomotion pattern that satisfies the control needs. Indeed speed control in hexapod is achieved modulating both the oscillation frequency of the neural control units and the phase displacement among the legs. In this work we refer only to the latter strategy which efficiently produces a modulation of the robot speed. In addition, it is required that the learning of phase displacement should take place in an unsupervised manner. To this aim, a particular neural network, known as *Motor Map*, was used. This is a generalization of the Self Organizing Feature Maps introduced by Kohonen. Here specific characteristics of the input patterns are mirrored in specific topological areas of the responding neurons: the space of the peculiar input pattern feature is mapped into the spatial location of the corresponding neural activity [29, 30]. This interesting potentiality can be well exploited for motion pattern generation, leading to the introduction of the *Motor Maps* (MMs) [31, 32]. Here the location of the neural activity within the Kohonen layer is able to produce a trainable weighted excitation, able to generate a motion which best matches an expected *reward signal*. Two layers are so considered: the Kohonen layer, devoted to the storage of learnable input weights and giving rise to a self organized topographic map, and the motor layer, where trainable output weights associate a suitable control signal to each input. The plastic characteristics of the Kohonen layer should also be preserved in the assignment of output values, so the learning phase deals with updating both the input and the output weights.

This is an extension of the winner-take-all algorithm. Once defined the dimension of the topographic map, typically by a trial and error method, and once randomly initialized the input and output weights, a Reward function is defined, which will guide the overall learning phase. At each learning step, the neuron  $q$  which best matches the pattern given as input is selected as the winning neuron.

Its output weight is used to perform the following *perturbed* control action  $A_q$ :

$$A_q = w_{q,out} + a_q \lambda \quad (5.9)$$

where  $w_{q,out}$  is the output weight of the winner neuron  $q$ ,  $a_q$  is a parameter determining the mean value of the search step for the neuron  $q$ , and  $\lambda$  is a Gaussian random variable with a zero mean. This is a way to guarantee a random search for possible solutions. Then the increase for the delta Reward Function (*DRF*) is computed and, if this value exceeds the average increase  $b_q$  gained at the neuron  $q$ , the weight update is performed; otherwise this step is skipped. The mean increase in the reward function is updated as follows:

$$b_q(new) = b_q(old) + \rho(DRF - b_q(old)) \quad (5.10)$$

where  $\rho$  is a positive value. Moreover,  $a_q$  is decreased as more and more experience is gained (this holds for the winner neuron and for the neighboring neurons), according to the following rule:



$$a_i(new) = a_i(old) + \eta_a \xi_a (a - a_i(old)) \quad (5.11)$$

where  $i$  indicates the generic neuron to be updated (the winner and its neighbors),  $a$  is a threshold the search step should converge to, and  $\eta_a$  is the learning rate, whereas  $\xi_a$  takes into account the fact that the parameters of the neurons to be updated are varied by different amounts, defining the extent and the shape of the neighborhood. If  $DRF > b_q$ , the weights of the winner neuron and those of its neighbors are updated following the rule:

$$\begin{cases} w_{i,in}(new) = w_{i,in}(old) + \eta \xi (v - w_{i,in}(old)) \\ w_{i,out}(new) = w_{i,out}(old) + \eta \xi (A - w_{i,out}(old)) \end{cases} \quad (5.12)$$

where  $\eta$  is the learning rate,  $\xi$ ,  $v$ ,  $w_{in}$ , and  $w_{out}$  are the neighborhood function, the input pattern, the input weights and the output weights, respectively. The subscript takes into account the neighborhood of the winner neuron.

The steps involving Eqs. 5.9–5.12 are repeated. If one wishes to preserve a residual plasticity for a later re-adaptation, by choosing  $a \neq 0$  in Eq. 5.11, learning is always active.

The idea proposed in this work is to use a hybrid approach joining the real time computation of RD-CNNs for generating stable locomotion patterns and a Motor Map as a high layer controller on the CNN-CPG, at the aim to control the speed of the robot, adapting the phase coordination among the legs. The MM input layer receives both the actual speed of the robot and the target speed used to evaluate the actual error. Each neuron provides a control law that modifies the phase displacement between the legs of the system.

The strategy adopted for the unsupervised modulation of the phases starts from considering the three stereotyped gaits generally adopted by hexapods and reported, in terms of phase displacements, in Table 5.1. These vary around the tripod gait and are able to maintain both static and dynamic stability. Here they are implemented by considering the front left leg  $L1$  as the reference leg and imposing a fixed phase relation between the front legs ( $\phi_{L1,L1} = 0^\circ$ ;  $\phi_{L1,R1} = 180^\circ$ ) for any of the selected gaits. From the inspection of Table 5.1 it emerges that, for example, a migration from gait G1 to G2 would imply a variation  $\delta\phi_{L1,L2} = \delta\phi_{L1,R2} = -30^\circ$  and a variation  $\delta\phi_{L1,L3} = \delta\phi_{L1,R3} = -60^\circ$ . Moreover, referring to Fig. 5.1, the network of command neurons was designed to leave the possibility to impose the oscillation phase of each leg independent on that one of the others. This is also allowed from the theoretical results, discussed previously, which enable any imposed phase dis-

**Table 5.1** Phase relation within legs in stereotyped locomotion gaits

Gait type	Speed (bodylength/s)	$\phi_{L1,L1}$	$\phi_{L1,R1}$	$\phi_{L1,L2}$	$\phi_{L1,R2}$	$\phi_{L1,L3}$	$\phi_{L1,R3}$
G1	0.6	0	180°	270°	90°	180°	360°
G2	0.73	0	180°	240°	60°	120°	300°
G3	0.77	0	180°	180°	0°	0°	180°

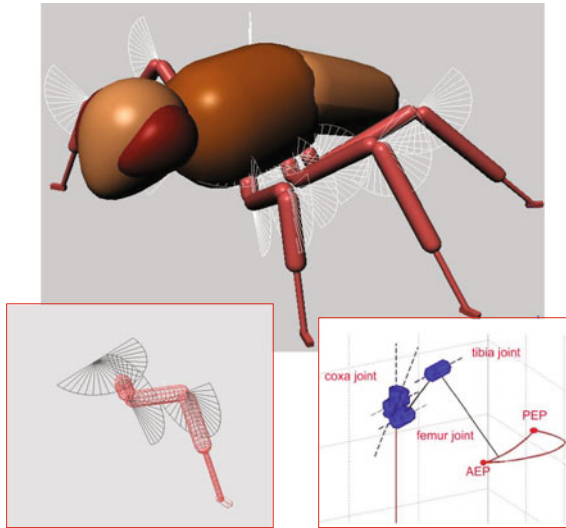
**Table 5.2** Parameters used for the motor map-based architecture designed to solve a speed control task

Parameters	Value
$\lambda$	$[-40, 40]$
$\eta$	$0.2 + 0.8 * e^{(-0.6t)}$
$a_q$	0.8
$b_q$	0
$a_{thr}$	0.2

placement among the legs to be reached exponentially. Of course, phase stability does not imply the dynamic stability of the gait. For this reason we preferred to move within the set of stable gaits  $G1, G2, G3$ , allowing the MM to find the most suitable phases to attain the desired speed. Within this plethora of gaits it is envisaged that a given phase imposed to a specific leg does not affect the phase associated with any other leg; to this aim the neurons belonging to the central backbone are synchronized and the phase displacements are imposed only to the connections from the backbone neurons to the outer cells. In this way, referring to Table 5.1, it holds, for example:  $\phi_{L1,R2} = \phi_{B2,R2}$ ,  $\phi_{L1,L3} = \phi_{B3,L3}$ , and so on. This symmetry in the phase modulation between the right and left side of the structure leads to adopt only two output trainable weights for the MM. Also the MM acts by adding increments in the phase displacement among the legs: this is preferable over sharply imposing absolute phases, that could imply potential loss of stability. In details, one of the two output weights represents the phase modulation  $\delta\phi_{B2,L2}$ , (which will also be imposed to  $\delta\phi_{B2,R2}$  for symmetry), the other stands for  $\delta\phi_{B3,L3} = \delta\phi_{B3,R3}$ . To find the suitable output weights a reward function that takes into account the speed error is considered and used to guide the learning process:  $R = -(speed_{actual} - speed_{ref})^2$ . The learning phase will find the weight set leading  $R$  as much as possible close to zero. The MM parameters used in the following simulations are reported in Table 5.2.

## 5.4 Dynamic Simulator

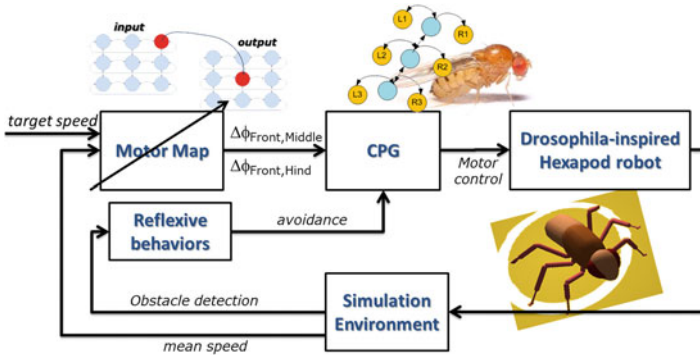
To validate the approach proposed, a dynamic simulator represents a suitable platform where these cognitive bio-inspired structures can be simulated, in view of being implemented in real robot prototypes for real life scenarios. The simulator is written in C++, and uses Open Dynamics Engine (ODE) as a physics engine to simulate dynamics and collision detection, and Open Scene Graph (OSG) as high performance 3D rendering engine [1, 33]. The main novelty of this approach consists in the extreme extensibility to introduce models. In fact, to import robot models in the simulator, a procedure was developed which starts from models realized in 3D Studio MAX and provides, using NVIDIA Physics Plugin for 3D Studio MAX, a COLLADA (COLLABorative Design Activity) description of the model to permit the correct transport in the simulated environment. In this way, the possibility to simulate own



**Fig. 5.2** Dynamical model of the *Drosophila*-inspired hexapod robot; the lower left panel reports the details of the operation range for the three joints of the left front leg, whereas the right bottom panel reports the kinematic simulation of the hind leg, showing the motion of the feet, cycling among the Anterior Extreme Position (AEP) and the Posterior Extreme Position (PEP)

environments and robots is guaranteed. The dynamical model of the *Drosophila*-inspired robot is shown in Fig. 5.2 where the typical asymmetric design and sprawled posture is evident. The structure includes a total of  $18^\circ$  of freedom, three for each leg. The legs, like in the real insect, are different in shapes and functionalities. Figure 5.2 reports also the operation ranges for the different leg joints for the front leg, as well as the rotation axes for the leg actuators, drawn by the corresponding Matlab Kinematic simulation. Here the trajectory spanned by the tip of the hind leg is also reported. The emerging limit cycle is the result of the application of the three *Primitive Functions* arising by the motor neuron net in Fig. 5.1. The robot is equipped with distance sensors, placed on the head and ground contact sensors located on the tip of each leg. The dimension is in scale with the biological counterpart with a body length of about 2.5 mm.

A block diagram showing the complete control structure is reported in Fig. 5.3. The CPG is able to generate the locomotion patterns of the robot depending on the control signals coming from the Motor Map for the speed control and from a reflexive behavior path that can trigger obstacles avoidance behaviors if an object is detected from the sensors equipped on the robot.



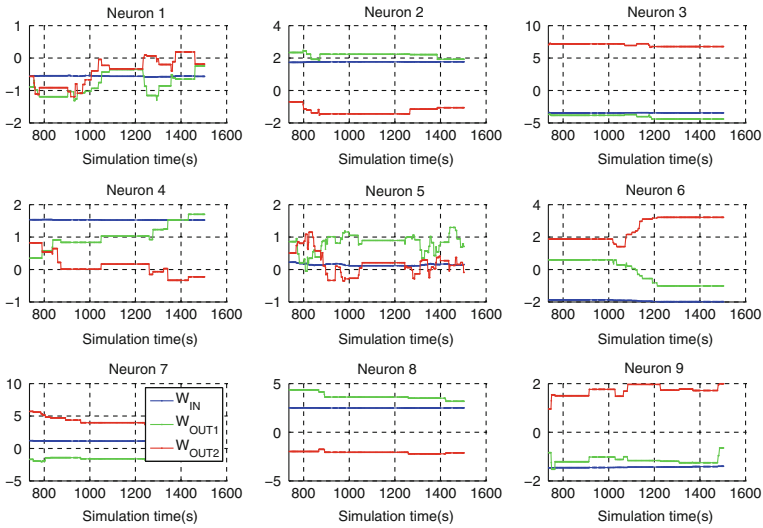
**Fig. 5.3** Block diagram of the control system used to guide the locomotion of the hexapod structure. The CPG motor signals are modulated by the inputs provided by the Motor Maps for speed control and by the reflexive path that takes the lead in presence of obstacle to be avoided. The MM receives in input the reference speed and the current speed whereas the reflexive behavior block is elicited in presence of an obstacle detected through distance sensors. The MM acts on the CPG by changing the phase displacement of the middle and hind legs whereas the reflexive block inhibits the learning procedure for the MM and activates a turning strategy

## 5.5 Simulation Results

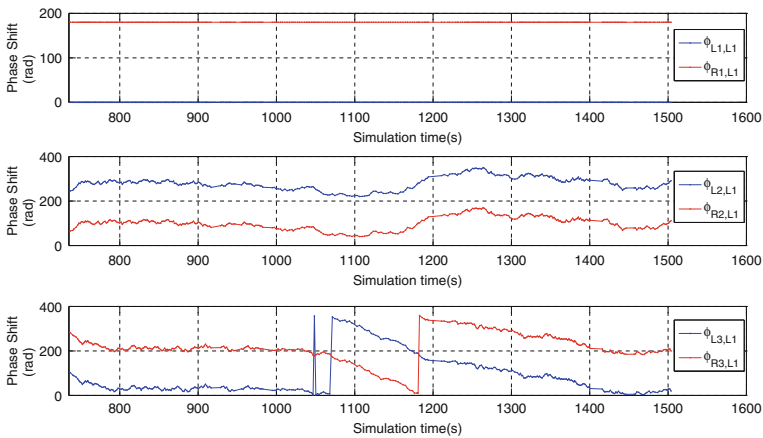
Analyzing the learning process in the proposed architecture we considered to start from gait  $G_2$  as initial configuration. The input related to the target speed, used for the MM neurons, is important when a time varying speed profile should be followed. For a first trial we considered a fixed speed, limiting the input to the speed error. This solution allowed to use a reduced number of neurons to create the map. In the following simulations a  $3 \times 3$  lattice was adopted in the Kohonen layer of the MM. To evaluate the mean speed of the robot in the dynamic simulator, soon after imposing a new phase configuration, a complete stepping cycle is performed to leave the dynamics to reach a steady state: then the displacement over the two subsequent stepping cycles is evaluated to have a consistent speed result.

The stepping time imposed by the CPG to each leg is about 1.5 s. The arena used during the learning process is limited by walls. When the robot detects an obstacle (or the arena walls) with the distance sensor placed on the head, a turning strategy is applied. During this avoidance behavior the speed evaluation is stopped and the learning process waits until the procedure is completed.

The time evolution of the input and output weights after an initial transient is shown in Fig. 5.4. It can be noticed that each neuron specializes to incrementally reach the desired solution: the topological arrangement of the Kohonen layer leads to the specialization of each neuron to a specific range of the input value. The output weights affect the leg phase displacement and the trend is shown in Fig. 5.5 where the anterior left leg ( $L_1$ ) that is directly connected with the backbone with zero phase is considered as reference for all the other legs. The phase relation for the anterior



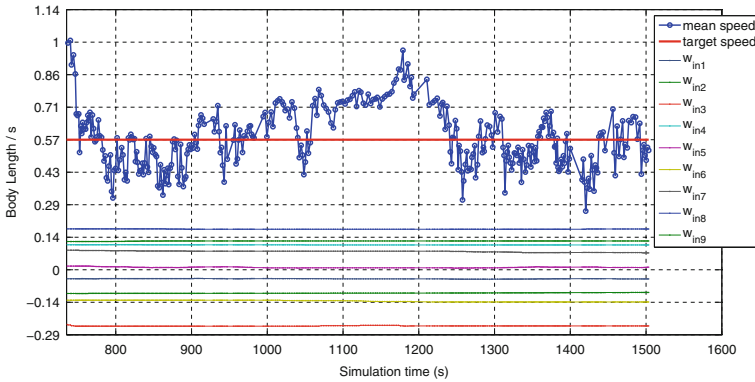
**Fig. 5.4** Time evolution of the input and output weights of the MM after an initial transient



**Fig. 5.5** Phase displacement in middle and hind legs obtained during learning

legs is not affected by adaptation, as explained above, and remains unchanged during the learning process.

The performance of the robot when autonomously learning to follow the reference speed is reported in Fig. 5.6. The initial speed of about 1 bodylength/s changes in time to reach a steady state solution of about 0.57 bodylength/s that was provided as reference speed. The same figure reports also the trend of the input weights related to each of the Kohonen neurons. Their values, after an initial transient, reach their

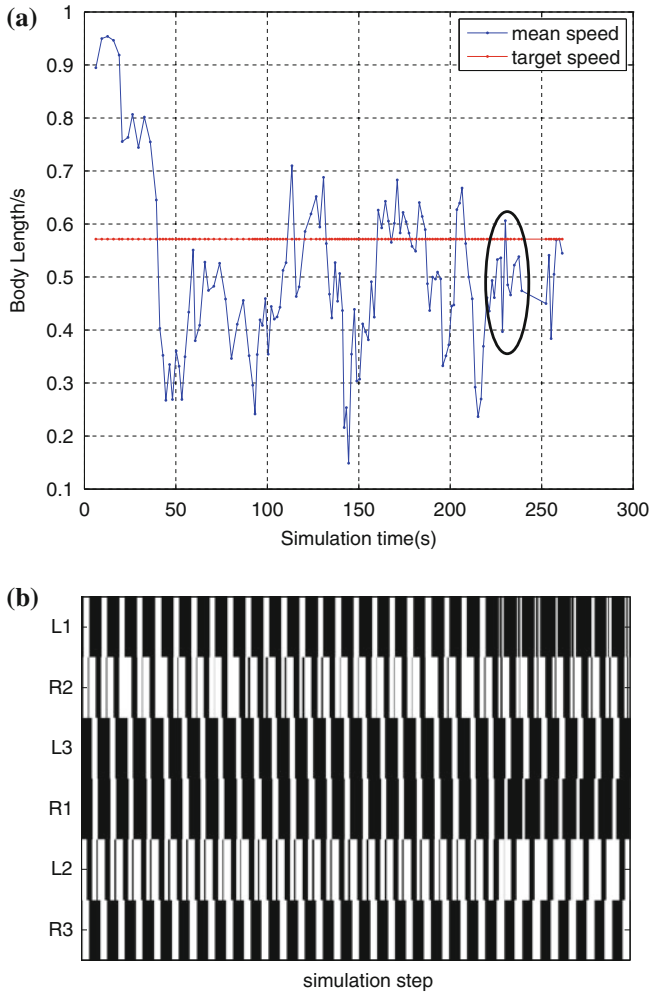


**Fig. 5.6** Time evolution of the input weights and the robot speed. The target speed is 0.57 bodylength/s and the robot after a transient converges toward this value. Each *circle* indicates a measured speed, the distance between each marker is not constant because the speed evaluation is stopped during the obstacle avoidance behavior

steady state distribution which encodes the unsupervised, topological clustering of the input velocity space.

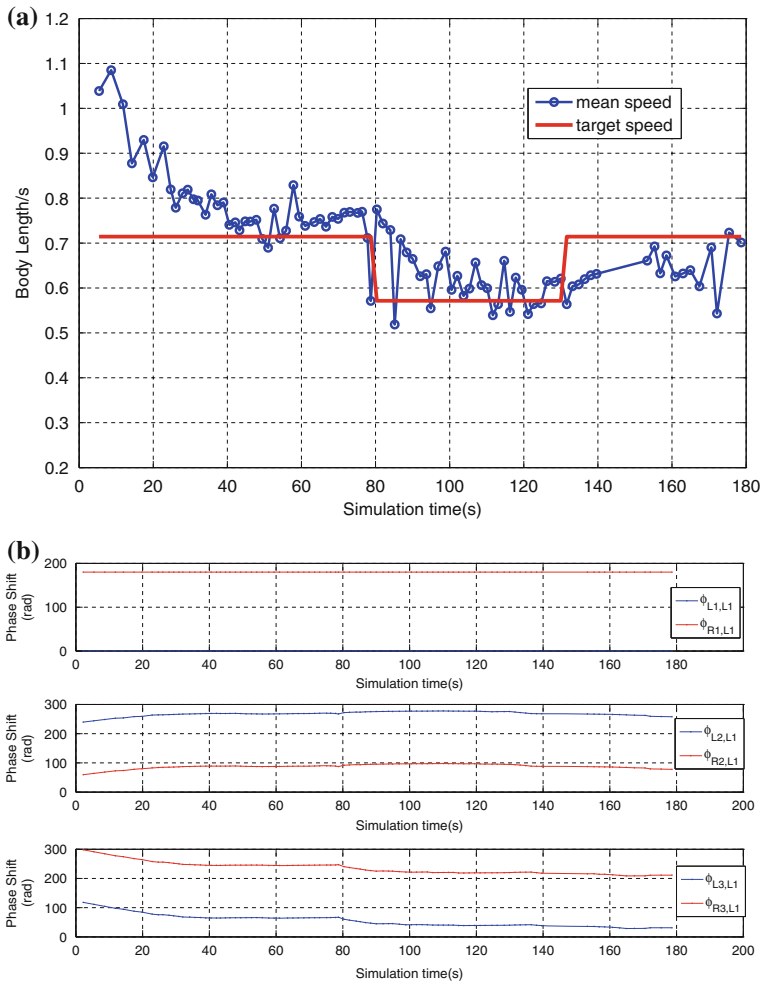
The locomotion gait can be analyzed using the stepping diagram that reports the stance and swing phase for each leg. The experiment reported in Fig. 5.7 shows a typical simulation starting from the *G2* gait configuration: during learning, the imposed phase from the MM controller onto the legs results in a gait transition, appreciated through the modulation of the stepping diagram of the robot. This diagram is recorded acquiring information from the ground contact sensor located on the tip of each leg. In this way it is possible to better understand the phase relation between legs, also taking into account the noise intrinsically present in a dynamic environment. Learning causes the phase locking between legs to change in time, searching for a suitable configuration that matches the desired speed value. It should be noticed that the application of the proposed strategy to a dynamic simulator is really similar to the outcome in a real experiment. Learning the complex map between the error speed and the corresponding phase displacement to be imposed passes through a series of unsuccessful trials where, for a given speed error currently provided in input, a trial phase is applied to the leg: if this choice is a failure, i.e. it does not contribute to an increase in the reward function, the simulator cannot come back to the previous stage. Instead it continues with the applied phase looking for a future rewarding choice.

An important element to evaluate the system performance is to test the control architecture: after 1,500s of simulation (i.e. in this time window the learning cycles in the motor map oscillate between 200 and 300) the weight adaptation was frozen and the network performance was tested. An interesting test is shown in Fig. 5.8, where a time varying speed profile is given as target to the robot. The robot easily reaches the first assigned speed of 0.71 bodylength/s and, in a short time, is able to



**Fig. 5.7** Stepping diagram obtained looking to the ground contact sensors placed in each leg tip of the simulated *Drosophila*-inspired robot. **a** The system starts from a phase configuration next to the gait *G2* and evolves in a series of free gaits trying to reach the desired speed. **b** In the stepping diagram that corresponds to the area outlined in the panel **a** the stance phase is shown in *black*, the swing phase in *white* and the legs are labeled as *left* (L) and *right* (R) and from *front* to *back* with numbers

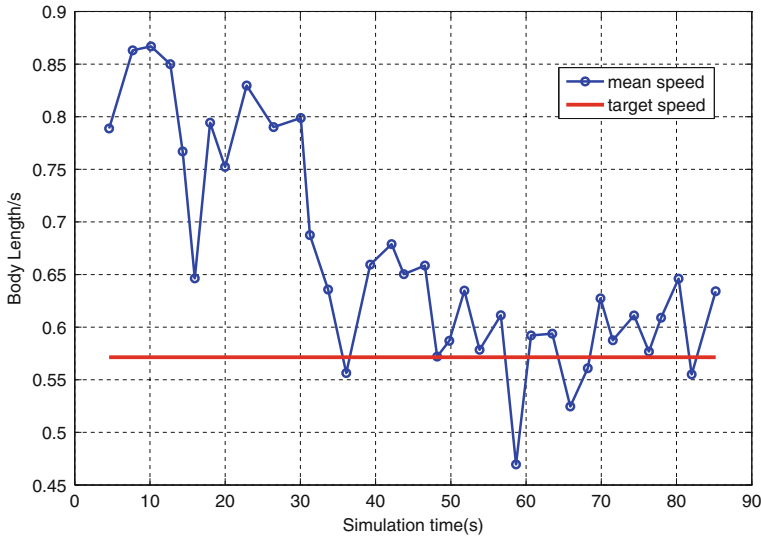
readapt the robot gait to reduce the speed to 0.57 bodylength/s to finally grow up again to the previous reference speed. The phase adaptation in time is also shown: during the testing phase the leg phases change due to the incremental effect of the output weights that however remain unchanged in time. As a classical controller, the learned non linear control law is governed by the varying speed error, provided at the network input.



**Fig. 5.8** Testing phase with different target speed values. **a** Time evolution of the robot speed. The target speed changes from 0.71 to 0.57 bodylength/s and the robot after a transient, converges toward the reference value in time. **b** Trend of the phase displacement among the robot legs during the testing phase

The learning is robust also to different initial configurations of the system walking gait (see Table 5.1). Using the same network already learned as previously presented, the robot is able to reach the target speed also starting from a different gait (i.e. *G2* gait) not used as starting condition during the learning iterations. Figure 5.9 shows the followed speed profile that converges to a target value of 0.57 bodylength/s.



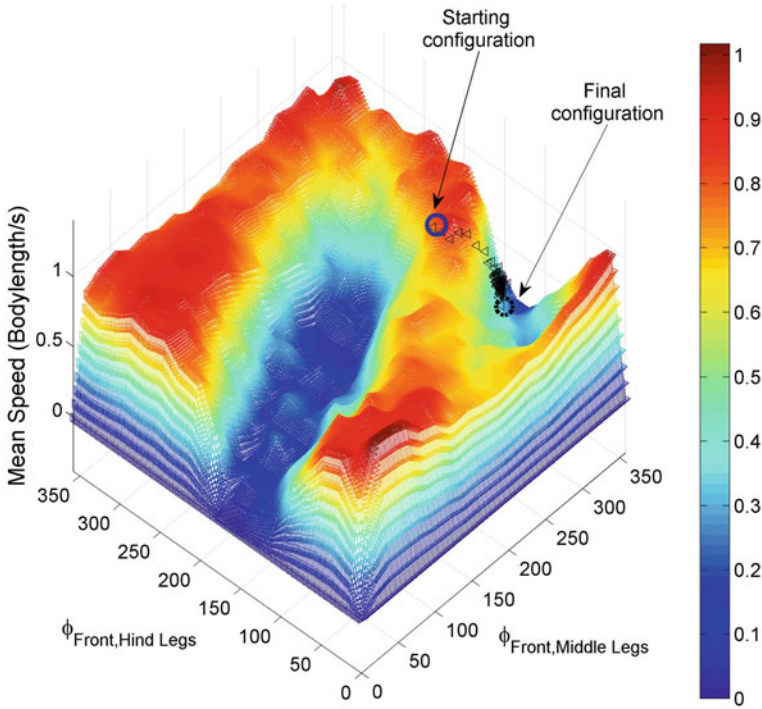


**Fig. 5.9** Time evolution of the robot speed. The target speed is 0.57 bodylength/s and the robot starts from a medium gait configuration that differs from the learning phase where a slow gait was considered as starting locomotion pattern

At the aim to evaluate the efficiency of the learned control law, the a-posteriori evaluation of the searching space in the phase domain is significant to enhance the important role of the MM speed controller. Figure 5.10 gives a qualitative idea about the searching space where the different phase configurations of the legs are related with the associated robot speed. The searching domain is complex and a non-linear mapping provided by the MM is needed to find a suitable solution to solve the speed control problem. To better illustrate the movement on the map the trajectory (in the space of the imposed phases) is also reported. Starting from a given initial configuration, the robot reaches the target speed finding the suitable path within the highly non-linear searching space.

## 5.6 Conclusions

Modeling neural structures is an important link in a chain connecting together Biologists and Engineers that cooperate to improve the knowledge on the neural mechanisms generating our behaviors and to develop new robots able to show adaptive capabilities similarly to their biological counterparts. A cellular automata approach to this problem, extended to the analog time domain with the CNNs, gives an interesting prospective because it easily allows theoretical analysis concerning stability issue, and also thanks to the mainly local connections between neurons, it creates a short-cut to the hardware implementation: different solutions are available using



**Fig. 5.10** Map of the searching space where, for each phase pair for middle and hind legs, the corresponding speed value is considered (i.e. reported in the z-axis). The trajectory followed by the system in the phase domain during a testing simulation is also reported: the starting point speed is about 0.89 bodylength/s, the robot reduced it reaching a speed of 0.46 bodylength/s following the learned trajectory in the phase space thanks to the MM neurons action

both micro-controller and dedicated integrated circuits. A Central Pattern Generator has been designed to generate the locomotion patterns for an asymmetric hexapod robot inspired by the *Drosophila melanogaster*. The integration of a neural controller based on Motor Maps allowed to adaptively control the robot speed acting on the CPG parameters. The reported results showed that the dynamically simulated robot was able to follow a desired speed profile incrementally adapting the phase displacement between legs. The neural controller shows interesting generalization capabilities to efficiently respond to novel initial conditions and time-varying speed profiles. This approach to adaptive speed control has to be considered as a brick within the much wider design of an insect brain computational model, where adaptive locomotion capabilities are required to show complex cognitive skills for the next generation of adaptive intelligent machines more and more mimicking their biological counterpart.

**Acknowledgments** This work was supported by EU Project EMICAB, grant no. 270182.

## References

1. Arena, P., Patané, L. (eds.): Spatial temporal patterns for action-oriented perception in roving Robots II. In: *An Insect Brain Computational Model Springer Series, Cognitive Systems Monographs*, vol. 21, Springer, Berlin (2014)
2. Arena, P., Patané, L. (eds.): Spatial Temporal Patterns for Action-Oriented Perception in Roving Robots. In: *Series, Cognitive Systems Monographs*, vol. 1, Springer, Berlin (2009)
3. Arena, P., Fortuna, L., Lombardo, D., Patané, L.: Perception for action: dynamic spatiotemporal patterns applied on a roving robot. *Adapt. Behav.* **16**(2–3), 104–121 (2008)
4. Arena, P., Fortuna, L., Frasca, M., Lombardo, D., Patané, L., Crucitti, P.: Turing patterns in RD-CNNs for the emergence of perceptual states in roving robots. *Int. J. Bifurcat. Chaos* **17**(1), 107–127 (2007)
5. Arena, P., Fortuna, L., Lombardo, D., Patané, L., Velarde, M.G.: The winnerless competition paradigm in cellular nonlinear networks: models and applications. *Int. J. Circuit Theory Appl.* **37**(4), 505–528 (2009)
6. Arena, P., De Fiore, S., Fortuna, L., Nicolosi, L., Patané, L., Vagliasindi, G.: Visual Homing: experimental results on an autonomous robot. In: *Proceedings of 18th European Conference on Circuit Theory and Design (ECCTD 07)* Seville, Spain (2007)
7. Arena, P., De Fiore, S., Patané, L., Termini, P.S., Strauss, R.: Visual learning in Drosophila: application on a roving robot and comparisons. In: *Proceedings of 5th SPIE's International Symposium on Microtechnologies Prague, Czech Republic* (2011)
8. Mronz, M., Strauss, R.: Visual Motion integration controls attractiveness of objects in walking flies and a mobile robot. In: *Proceedings of International Conference on Intelligent Robots and Systems*, pp. 3559–3564, Nice, France (2008)
9. Arena, P., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: CPG-MTA implementation for locomotion control. In: *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 4102–4105 (2005)
10. Arena, P., Fortuna, L., Frasca, M., Patané, L., Pollino, M.: An autonomous mini-hexapod robot controlled through a CNN-based CPG VLSI chip. In: *Proceedings of 10th IEEE International Workshop on Cellular Neural Networks and their Applications*, pp. 401–406, Istanbul, Turkey (2006)
11. Arena, P., Fortuna, L., Frasca, M., Patané, L.: Sensory feedback in CNN-based central pattern generators. *Int. J. Neural Syst.* **13**(6), 349–362 (2003)
12. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks* **21**, 642–653 (2008)
13. Arena, P., Patané, L.: Simple sensors provide inputs for cognitive robots. *IEEE Instrum. Meas. Mag* **12**(3), 13–20 (2009)
14. Steingrube, S., Timme, M., Worgotter, F., Manoonpong, P.: Self-organized adaptation of a simple neural circuit enables complex robot behaviour. *Nature Phys.* **6**, 224–230 (2010)
15. Pavone, M., Arena, P., Fortuna, L., Frasca, M., Patané, L.: Climbing obstacle in bio-robots via CNN and adaptive attitude control. *Int. J. Circuit Theory Appl.* **34**(1), 109–125 (2006)
16. Arena, P., Fortuna, L., Frasca, M.: Attitude control in walking hexapod robots: an analogic spatio-temporal approach. *Int. J. Circuit Theory Appl.* **30**(2–3), 349–362 (2002)
17. Seo, K., Slotine, J.E.: Models for Global Synchronization in CPG-based Locomotion. In: *Proceedings of IEEE ICRA* (2007)
18. Arena, P., De Fiore, S., Patané, L.: Cellular nonlinear networks for the emergence of perceptual states: application to robot navigation control. *Neural Networks* **22**(5–6), 801–811 (2009). ISSN 08936080 (special issue: advance in neural networks research)
19. Arena, P., Fortuna, L., De Fiore, S., Frasca, M., Patané, L.: Perception-action map learning in controlled multiscroll system applied to robot navigation. *Chaos* **18**(043119), 1–16 (2008)
20. Arena, P., Fortuna, L., Frasca, M., Sicurella, G. (eds.): An adaptive. Self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *Syst. Man Cybern. Part B* **34**(4), 1823–1837 (2004)

21. Frasca, M., Arena, P., Fortuna, L.: Bio-inspired emergent control of locomotion systems, p. 48. World Scientific, Singapore (2004) (Serie A)
22. Manganaro, G., Arena, P., Fortuna, L.: Cellular neural networks: chaos, complexity, and VLSI processing. In: *Advanced Microelectronics*, vol. 1, Springer, Berlin (1999)
23. Arena, E., Arena, P., Patané, L.: Efficient hexapodal locomotion control based on flow-invariant subspaces. In: *Proceedings of 18th World Congress of the International Federation of Automatic Control (IFAC)*, Milan, Italy (2011)
24. Arena, E., Arena, P., Patané, L.: Frequency-driven gait control in a central pattern generator. In: *Proceedings of 1st International Conference on Applied Bionics and Biomechanics ICABB-2010*, Venice, Italy (2010)
25. Arena, E., Arena, P., Patané, L.: Modelling stepping strategies for steering in insects. In: *Proceedings of XXI Italian Workshop on Neural Networks* (2011)
26. Baek, M., Mann, R.S.: Lineage and birth date specify motor neuron targeting and dendritic architecture in adult *Drosophila*. *J. Neurosci.* **29**(21), 6904–6916 (2009)
27. Edwards, D.H., Heitler, W.J., Krasne, F.B.: Fifty years of a command neuron: the neurobiology of escape behavior in the crayfish. *Trends Neurosci.* **22**(4), 153–161 (1999)
28. Arena, E., Arena, P., Patané, L.: CPG-based locomotion generation in a *Drosophila* inspired legged robot. In: *Proceedings of Biorob 2012*, Roma, Italy (2012)
29. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Bio. Cybern.* **43**, 59–69 (1972)
30. Arena, P., De Fiore, S., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: Reactive navigation through multiscroll systems: from theory to real-time implementation. *Auton. Robot.* **25**(1–2), 123–146 (2008)
31. Ritter, H., Schulten, K.: Kohonen's self-organizing maps: exploring their computational capabilities. In: *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, San Diego, CA, 109–116 (1988)
32. Schulten, K.: Theoretical biophysics of living systems. [http://www.ks.uiuc.edu/Services/Class/PHYS498TBP/spring2002/neuro\\_book.html](http://www.ks.uiuc.edu/Services/Class/PHYS498TBP/spring2002/neuro_book.html) (2002)
33. Arena, P., Cosentino, M., Patané, L., Vitanza, A.: SPARKRS4CS: a software/hardware framework for cognitive architectures. In: *Proceedings of 5th SPIE's International Symposium on Microtechnologies*, pp. 1–12. Czech Republic, Prague (2011)