Chapter 6
Perception for Action in Roving Robots: A Dynamical System Approach

Paolo Arena, Sebastiano De Fiore, and Luca Patané

6.1 Introduction

For a mobile robot working in the real world, the ability to interpret information coming from the environment is crucial, both for its survival and for its accomplishing tasks. The cognitive capabilities of a robotic system are defined by the way in which the information gathered from sensors are processed to produce a specific action or behavior. Two broad classes can be distinguished: the *cognitivist approach* based on symbolic information processing and the *emergent systems approach* that is directed to the application of dynamical systems connected to the principles of self-organization [39]. Among the numerous solutions proposed by researchers, a great part is located in between the two main approaches.

In the recent past, research was directed towards endowing a robot with capabilities of self-creating an internal representation of the environment. Experiments in real world differ from applications in structured environments because they are mostly dynamically changing, so that it is impossible to program robot behaviors only on the basis of a priori knowledge. Moreover, the control loop must be able to process the different stimuli coming from the environment in a time that must be compatible with the real time applications. To solve these open problems, research activities have been focused on suitable solutions obtained taking inspiration from nature and applying biological principles to develop new control systems for perception–action purposes.

Recently inside the research activity of the EU Project SPARK [37], a bio-inspired framework for action-oriented perception has been proposed [5, 13].

Perception is no longer considered as a stand-alone process, but as an holistic and synergetic one tightly connected to the motor and cognitive system [14]. Perception processes are indivisible from action: behavioral needs provide the context for the perceptual process, which, in turn, works out the information required for motion control.

In this work, the perceptual system, based on principles borrowed by insect neurobiology, nonlinear dynamics, and complex system theory, has been realized by means of chaotic dynamical systems controlled through a new technique called *Weak Chaos Control* (WCC).

WCC technique allows to create perceptual states that can be managed by the control system because it directly is related to the concept of embodiment and situatedness [38]. The biological principles that inspired the proposed approach are based on Freeman's theories. His approach recognizes the existence of internal (mental) motivationally driven pattern formation [16, 19]. Cerebral cortex processes information coming from objects identified in the environment from receptors by enrolling dedicated neural assemblies. These are nonlinear dynamical coupled systems whose collective dynamics constitutes the mental representation of the stimulus. Freeman and co-workers, in their extensive experimental studies on the dynamics of sensory processing in animals [16–18, 24], conceive a dynamical theory of perception. Through electroencephalogram (EEG), Freeman evaluated the action potentials in the olfactory bulb and he noticed that the potential waves showed a typical chaotic behavior. So he came to the conclusion that the emergence of a mental pattern, in correspondence to an incoming stimulus, is the result of a chaotic dynamics in the sensory cortex in cooperation with the limbic system that implements the supporting processes of intention and attention [16].

The application of chaotic models as basic blocks to reproduce adaptive behaviors [4, 11, 36] is an interesting aspect of the current research activity on this field. The idea is that the chaos provides the right properties in terms of stability and flexibility, needed by systems that evolve among different cognitive states. In particular, perceptual systems dynamically migrate among different attractors that represent the meaning of the sensory stimuli coming from the environment.

Moreover, other studies consider the role of noise as an added value to reactive systems that otherwise could not be able to escape from the deadlock situations produced by local minima [28]. Chaotic dynamics can further improve reactive system capabilities, in fact chaotic systems generate a wide variety of attractors that can be controlled guiding the transit from one to another. Freeman's studies lead to a model, called K-sets, of the chaotic dynamics observed in the cortical olfactory system. This model has been used as dynamic memory for robust classification and navigation control of roving robots [20–22, 24].

The architecture proposed here, taking into consideration the relevant principles previously underlined, is based on the control of chaotic dynamics to learn adaptive behaviors in roving robots. The main aim is to formalize a new method of chaos control applied to solve problems of perceptual state formation.

The WCC approach following these guidelines uses a chaos control technique applied to a multiscroll chaotic system [27]. Therefore, the WCC is a general technique that can be applied to several chaotic systems [7, 41]. All sensory signals are mapped as different potential reference dynamics used to control the chaotic system. This creates association between sensory information and a particular area located within the multiscroll phase plane, in a way that reflects the topological position of the robot within the sensed environment. Moreover, thanks to the addition of a learning stage (called in the following *Action Selection layer*), the technique allows to implicitly include within the robot control system the real robotic structure and dimensions, including the sensor position, thus situating the robot within the environment. The improvements provided by the Action Selection layer are shown in several works [5, 10, 25]. Here a bio-inspired adaptive structure, based on an unsupervised, reward-based method derived by the *Motor Map* paradigm, has been used. This network, inspired by the paradigm of Kohonen Nets [23], is able to plastically react to localized excitation by triggering a movement (like the motor cortex or the superior colliculus in the brain) [30].

The aim of this chapter is to present in detail the WCC navigation technique using the Motor Map paradigm to select the robot action. Moreover, the proposed navigation control technique, based on the action-oriented perception paradigm, has been implemented in the FPGA-based board and tested on a roving robot in a real environment.

In the next section we will describe the control architecture that reproduce the sensing–perception–action loop. Section 6.3 is devoted to illustrate the hardware used to implement the proposed architecture. In Sect. 6.4, the implementation is explained in more details, while the results are reported in Sect. 6.5.2. Finally, in Sect. 6.7, we will draw the conclusions and present the possible developments that we are currently investigating.

6.2 Control Architecture

The sensing–perception–action loop is at the basis of research in the cognitive system field. In this chapter we propose an implementation of the action-oriented perception paradigm applied to the navigation control of an autonomous roving robot.

A basic scheme of the architecture is shown in Fig. 6.1, where two main blocks can be distinguished:

Perceptual system creates an internal representation of the environment from the sensory inputs.

Action selection network learns the suitable action associated to each environmental situation on the basis of the task to be accomplished.

When a robot is placed in an unknown environment, it is subject to a huge amount of external stimuli. To explore the area avoiding obstacles, the robot, sensing the environment, can create an internal representation of the stimuli in relation to its body. Therefore, when no stimuli are perceived (i.e., there are no active sensors), the system evolves in a chaotic behavior and the robot continues to explore the environment performing a random action determined by the chaotic evolution of a dynamical system representing the core of the perceptual formation mechanism. When external stimuli are perceived, the dynamical system evolution is enslaved into low order dynamics that depends on the contribution of each active sensor. The behavior that

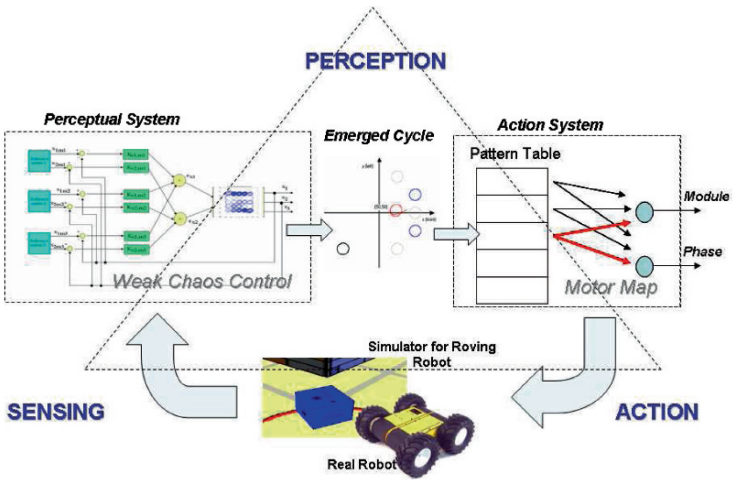


Fig. 6.1 Control architecture used to implement the sensing-perception-action loop on a roving robot

correspond to this perception will plastically depend on the characteristics emerging dynamics, as a result of a learning algorithm. The learning mechanism is here driven by a Reward Function (RF), designed on the basis of the mission to be accomplished (in our case navigation tasks). In the following sections, the main blocks are described in details.

6.2.1 Perceptual System

The core of this layer is the WCC method [6]. The crucial advantage of this approach is the possibility to create a compact representation of the environment that can be used for real-time navigation purposes in mobile robots. Moreover, this method permits to handle with a great number of sensors. It emulates the perceptual processes of the brain in which specific cerebral patterns emerge depending on the perceived sensorial stimuli. To model this behavior, a chaotic system, proposed by Chen [27], has been used as a plastic layer in which perceptual states can emerge. The chaotic behavior of the Chen's multiscroll system can be enslaved to regular periodic patterns (i.e., emergence of perceptual states) by using the sensory stimuli as reference control signals. The control mechanism has been realized with a feedback on the state variables x and y controlled in order to follow the reference cycles. This multiscroll system can be viewed as a generalization of the Chua's double scroll attractor represented through saturated piecewise linear functions and of other circuits able to generate n-scrolls [29]. It is able to generate one-dimensional (1D) n-scrolls, two-dimensional (2D) $n \times m$-grid scrolls, or three-dimensional (3D) $n \times m \times l$-grid scroll chaotic attractors by using saturated function series. In this paper, a 2D multiscroll system has been chosen. It is described by the following differential equations [27]:

$$\begin{cases} \dot{x} = y - \frac{d_2}{b} f_1(y; k_2; h_2; p_2, q_2), \\ \dot{y} = z, \\ \dot{z} = -ax - by - cz + d_1 f_1(x; k_1; h_1; p_1, q_1) \\ \qquad + d_2 f_1(y; k_2; h_2; p_2, q_2), \end{cases} \tag{6.1}$$

where the following so-called saturated function series (PWL) $f_1(x; k_j; h_j; p_j, q_j)$ has been used:

$$f_1(x; k_j; h_j; p_j; q_j) = \sum_{i=-p_j}^{q_j} g_i(x; k_j; h_j), \tag{6.2}$$

where $k_j > 0$ is the slope of the saturated function, $h_j > 2$ is called *saturated delay time*, p_j and q_j are positive integers, and

$$g_i(x; k_j; h_j) = \begin{cases} 2k_j & \text{if } x > ih_j + 1, \\ k_j(x - ih_j) + k_j & \text{if } |x - ih_j| \leq 1, \\ 0 & \text{if } x < ih_j - 1, \end{cases}$$

$$g_{-i}(x; k_j; h_j) = \begin{cases} 0 & \text{if } x > -ih_j + 1, \\ k_j(x + ih_j) - k_j & \text{if } |x + ih_j| \leq 1, \\ -2k_j & \text{if } x < -ih_j - 1. \end{cases}$$

System (6.1) can generate a grid of $(p_1 + q_1 + 2) * (p_2 + q_2 + 2)$ *scroll attractors*. Parameters p_1 (p_2) and q_1 (q_2) control the number of *scroll attractors* in the positive and negative direction of the variable x (y), respectively. The parameters used in the following ($a = b = c = d_1 = d_2 = 0.7$, $k_1 = k_2 = 50$, $h_1 = h_2 = 100$, $p_1 = p_2 = 1$, $q_1 = q_2 = 2$) have been chosen according to the guidelines introduced in [27] to generate a 2D 5×5 grid of scroll attractors. An example of the chaotic dynamics of system (6.1) is given in Fig. 6.2, also where a 3D grid of scrolls is shown.

In our approach, the perceptual system is represented by the multiscroll attractor of (6.1), while sensorial stimuli interact with the system through periodic inputs that can modify the internal chaotic behavior. As one of the main characteristics of perceptive systems is that sensorial stimuli strongly influence the spatial-temporal dynamics of the internal state, a suitable scheme to control the chaotic behavior of the multiscroll system on the basis of sensorial stimuli should be adopted.

Chaos control refers to a process wherein a tiny perturbation is applied to a chaotic system to realize a desirable behavior (e.g., chaotic, periodic, and others). Several techniques have been developed for the control of chaos [15].

A commonly used chaos control strategy is to select the desirable behavior among the unstable periodic orbits embedded in the system. The stabilization of one of the unstable periodic orbits can be obtained by applying a small perturbation when the chaotic orbit passes close to the desired behavior.

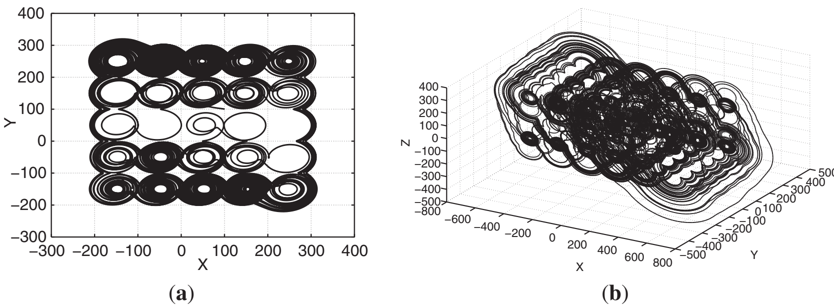


Fig. 6.2 (**a**) Projection of the 5×5 grid of *scroll attractors* in the plane *x*-*y*. (**b**) 3D-scroll attractor, 5×5×5 grid of scrolls

In general, the strategies for the chaos control can be divided in two classes: closed loop and open loop. The first class includes those methods that select the perturbation based on a knowledge of the state variables and oriented to control a prescribed dynamics. The two most important feedback methods are the Ott–Grebogi–York (OGY) [15,31] approach and the Pyragas technique [32]. The second class includes those strategies that consider the effect of external perturbations on the evolution of the system. In the OGY's method, the changes of the parameter are discrete in time as this method is based on the use of Poincar maps.

In view of our application, a continuous-time technique like the Pyragas's method is a suitable choice [32]. In this method [32, 33], the following model is taken into account:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = P(y,x) + F(t), \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = Q(y,x), \tag{6.3}$$

where y is the output of the system (i.e., a subset of the state variables) and the vector x describes the remaining state variables of the system. $F(t)$ is the additive feedback perturbation that forces the chaotic system to follow the desired dynamics. Pyragas [32, 33] introduced two different methods of permanent control in the form of feedback. In the first method, that is used here, $F(t)$ assumes the following form:

$$F(t) = K[\widehat{y}(t) - y(t)], \tag{6.4}$$

where $\widehat{y}$ represents the external input (i.e. the desired dynamics), and K represents a vector of experimental adjustable weights (adaptive control). The method can be employed to stabilize the unstable orbits endowed in the chaotic attractor reducing the high order dynamics of the chaotic system.

In the second method, the idea consists in substituting the external signal $\widehat{y}$ in (6.4) with the delayed output signal $y(t-\tau)$

$$F(t) = K[y(t-\tau) - y(t)], \tag{6.5}$$

where τ is a delay in time. This feedback performs the function of self-control.

In our case, a strategy based on (6.4) has been applied. The desired dynamics is provided by the periodic behavior associated with the sensorial stimuli. As more than one stimulus can be presented at the same time, the Pyragas method has been generalized to account for more than one external forcing.

Hence, the equations of the controlled multiscroll system can be written as follows:

$$\begin{cases} \dot{x} = y - \frac{d_2}{b} f_1(y;k_2;h_2;p_2,q_2) + \sum_i K_{x_i}(x_{r_i} - x) \\ \dot{y} = z + \sum_i K_{y_i}(y_{r_i} - y) \\ \dot{z} = -ax - by - cz + d_1 f_1(x;k_1;h_1;p_1,q_1) \\ \qquad + d_2 f_1(y;k_2;h_2;p_2,q_2), \end{cases} \tag{6.6}$$

where i is the number of external references acting on the system, x_{r_i}, y_{r_i} are the state variables of the reference circuits, which will be described in detail, and K_{x_i}, K_{y_i} represent the control gains. It can be noticed that the control acts only on the state variables x and y. The complete control scheme is shown in Fig. 6.3.

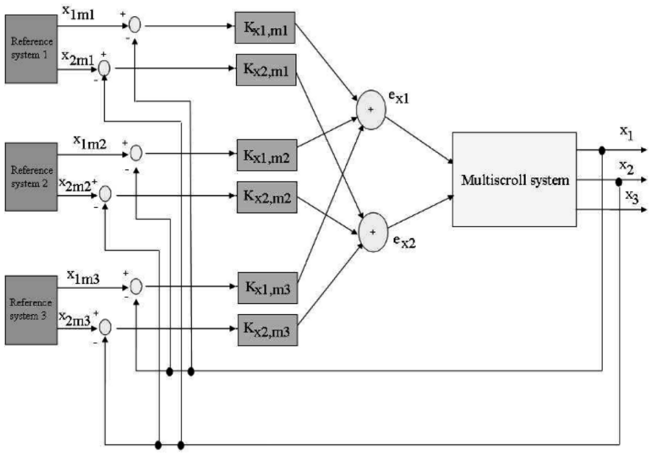


Fig. 6.3 The control scheme when three distinct reference signals (i.e., sensorial stimuli) are perceived by the multiscroll system

Each reference signal (x_{r_i}, y_{r_i}) is a periodic trajectory that represents a native cycle. It can be generated using a multiscroll system (6.1) with particular parameters ($a = b = c = 1$). The amplitude, pulse, and center position of the cycle depend on the initial conditions. The number of the multiscroll systems needed to generate the reference cycles is the same as the number of reference trajectories required. These reference signals can be built using a sinusoidal oscillator:

$$x_r(t) = A_{x_r} \sin(\omega_{xr} - \varphi_{x_r}) + x_{\text{off}}$$

$$y_r(t) = A_{y_r} \sin(\omega_{yr} - \varphi_{y_r}) + y_{\text{off}} \tag{6.7}$$

where $(x_{\text{off}}, y_{\text{off}})$ is the center of the reference cycle, ω is a frequency (in this work $\omega_{xr} = \omega_{yr} = 1$), φ_{x_r} and φ_{y_r} are the phases, A_{x_r} and A_{y_r} define the amplitude of the reference.

This solution allows to improve the performance of the control system: only the differential equations of the controlled system must be integrated, the reference trajectory are obtained using (6.7).

If the sensor stimuli are associated with the reference cycles, the WCC technique can be used to control robot navigation. A key point of this approach is that the reference cycle distribution in the phase plane *x*-*y* reflects the topological distribution of robot sensors; this contributes to include the geometrical embodiment of the robot itself within the environment. Moreover, the sensor range depicts the current robot operating space, which is dynamically encoded within the phase space of the multiscroll system. The link between reference signals and sensors are obtained through control gains K_{x_i} and K_{y_i}. The value of these control parameters are related to the amplitude of the sensory stimuli, and so a regular periodic pattern emerges as a function of the sensor readings.

Figure 6.4 shows an example of the association between sensors and reference signals used to control the chaotic system. Sensors with the same position on the robot are associated with reference cycles with the same position in the phase plane. Distance sensors are associated with the reference cycles that reflect their topological position. A similar strategy has been adopted for the target. When a target is within the range of robot visibility, it is considered as an obstacle located in a position symmetric with respect to the motion direction. The aiming action is guided by a reference cycle with a low gain, so that obstacle avoidance is a priority over reaching a target.

Each cycle that emerges from the control process (i.e., perceptual state) can be identified through its center position and shape. A code is then associated to each cycle and it is defined by the following parameters:

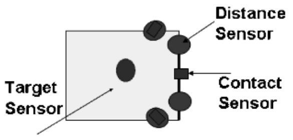
- x_q and y_q: the center position in the phase plane $x - y$
- $\bar{x}_q$: maximum variation of the state variable x within the emerged cycle
- $\bar{y}_q$: maximum variation of the state variable y within the emerged cycle

where q indicates the emerged cycle.

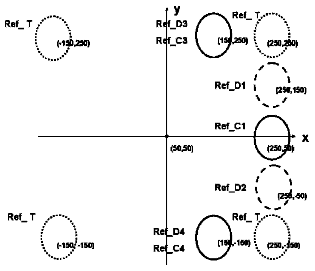
In this way, a few parameters provide an abstract and concise representation of the environment. To solve the robot navigation task, an action is performed by the robot according to the characteristics of the emerged pattern.

6.2.2 Action Selection Layer

The perceptual pattern obtained through the WCC technique is then processed by the action selection system (see Fig. 6.1).



(a)



(b)

Fig. 6.4 (**a**) Map of the sensors equipped on the robot. (**b**) Reference cycles Ref_i linked to the sensors. Contact sensors are associated to the same cycles as the corresponding distance sensors, if they are placed in the same position on the robot. The T_i reference cycles are associated to the target sensor. The target is considered as an obstacle located in a symmetrical position with respect to the motion direction

The action block establishes the association between the emerged cycle and the consequent robot action. An action consists of two elements:

$$action = (module, phase). \tag{6.8}$$

The module and phase of an action determine, respectively, the motion step and the rotation angle to be performed by the robot. To perform this task, the Motor Map (MM) paradigm was applied. MMs are suitable to control the behaviors of the robot in an unknown environment, because they are adaptive, unsupervised structures and are simple enough to allow for a real-time learning. The MM is composed by two layers: one (V) devoted to the storage of input weights and another (U) devoted to the output weights. V represents the actual state of the system which is to be controlled and U determines the required control action. This allows the map to perform tasks such as motor control. Formally, a MM can be defined as an array of neurons mapping the space of the input patterns into the space of the output actions:

$$\varphi : V \longrightarrow U. \tag{6.9}$$

The learning algorithm is the key point to obtain a spatial arrangement of both the input and output weight values of the map. This is achieved by considering an extension of the Kohonen algorithm. At each learning step, when a pattern is given as input, the winner neuron is identified: this is the neuron that best matches the input pattern. Then, a neighborhood of the winner neuron is considered and an update involving both the input and output weights for neurons belonging to this neighborhood is performed. The learning procedure is driven by a reward function that is defined on the basis of the final aim of the control process [35]. A MM although very efficient to be trained could be difficult to be implemented in hardware because of the high number of afferent and efferent weights. As this work is finalized to the application on real roving robot prototypes, a simplified version is adopted [5]. The first difference is that the relationship of proximity is not considered among the neurons. Another and more important difference is that the afferent layer is substituted by a pattern table. Each emerged cycle, identified by a code, is stored in the pattern table (if it is not yet present) when the pattern emerges for the first time. Each element in the pattern table contains the emerged cycle code and the number of iterations from its last occurrence (defined as *age*). If the pattern table is full, the new element will overwrite the one containing the code of the pattern least recently used (LRU), that is, the one with the highest *age* value. This table permits to synthesize the perception of the environment. Therefore, the winning neuron of the afferent layer is replaced by the element q of the pattern table that contains the last emerged cycle. Moreover, the efferent (output) layer is now constituted of two weights for each element of the pattern vector. The element q is connected to the weights w_{qm} and w_{qp}, which represent, respectively, module and phase of the action associated with the pattern q (A_q). At each step, the robot does not perform the exact action suggested by the weights of q (w_{qm} and w_{qp}), but the final action is

$$A_q = (A_q(\text{module}), A_q(\text{phase})) = (w_{qm} + a_{s_q}\lambda_1, w_{qp} + a_{s_q}\lambda_2), \tag{6.10}$$

where λ_1 and λ_2 are random variables uniformly distributed in the range $[-1;1]$. The parameter a_{s_q} limits the searching area. Every time the pattern q emerges, a_{s_q} is reduced to focus the action search in a smaller range so as to guarantee the convergence of the efferent weights. When there are no inputs, the perceptual core of the robot (the multiscroll system) behaves chaotically. This implies that there are no emerged cycles and no entries in the pattern table. In this case, the robot explores the environment and its action depends on the position of the centroid of the particular chaotic scroll shown by the system during the simulation step. Of course, the exploration phase can also be performed using a forward motion, that is, not considering the chaotic wandering.

The unsupervised learning mechanism that characterizes the MM algorithm is based on a reward function (RF). This is a fitness function and it is the unique information that permits to determine the effectiveness of an action. In a random foraging task, a suitable choice for the RF is

$$\text{RF} = -\sum_i \frac{k_i}{D_i^2} - h_D D_T - h_A|\phi_T|, \tag{6.11}$$

where D_i is the distance between the robot and the obstacle detected by the sensor i, D_T is the robot–target distance, ϕ_T is the angle between the direction of the longitudinal axis of the robot and the direction connecting robot and target, and k_i, h_D, and h_A are appropriate positive constants determined during the design phase [9, 12].

6.3 Hardware Devices

The proposed method for robot navigation was tested in a simulation environment showing good results [8]. The choices done during the design of the control architecture, oriented to a real-time hardware implementation, simplify the implementation of the strategy in a high performing hardware to be embedded on a roving robot.

The first step followed consisted in the definition of the hardware framework. The system should be able to acquire information coming from a distributed sensory system. Moreover, the computational power have to be sufficient to obtain control command compatible with a real-time application, allowing in the mean time the data logging for debugging and post-processing performance analysis.

The two main options for the controller are a digital signal processor (DSP) or a field programmable gate array (FPGA). A microcontroller is not taken into account due to the limited resource and performance that could compromise the scalability of the system.

DSPs are currently used in several fields of applications: communications, medical diagnostic equipment, military systems, audio and video equipment, and countless other products, becoming increasingly common in consumers' lives.

DSPs are a specialized form of microprocessor, while FPGAs are a form of highly configurable hardware. In the past, the use of digital signal processors was ubiquitous, but now, with the needs of many applications outstripping the processing capabilities of DSPs (measured in millions of instructions per second (MIPS)), the use of FPGAs is growing rapidly. Thus, the comparison between digital signal processors and FPGAs focuses on MIPS comparison, which, while certainly important, is not the only advantage of an FPGA. Equally important, and often overlooked, is the FPGAs inherent advantage in product reliability and maintainability. Therefore, the device chosen to implement the WCC approach is an FPGA.

The particular FPGA device adopted is a Stratix II EP2S60 produced by Altera, which integrates a soft embedded processor, the NiosII. This solution was adopted to combine in a single device both the hardware reconfigurability of FPGA devices and the programming simplicity of microcontrollers. NiosII is, in fact, a 32-bit RISC digital microprocessor with a clock frequency of 50 MHz. It is a configurable "soft-core" processor. Soft-core means that the CPU core is offered in "soft" design form (not fixed in silicon), so that its functionalities, like the peripheral number and type or the amount of memory, can be easily modified according to the specific task [2].

The first attempt was carried out by using a development board provide by Altera [2]. This preliminary architecture led to the design of a new FPGA-based board, called SPARK board. The SPARK board has been developed during the SPARK project [37] and has been designed to fulfill the requirements needed by the SPARK cognitive architecture [13].

The mobile robot used for the experiment is called Rover II (see Fig. 6.5). It is a classic four-wheel drive rover controlled through a differential drive system by Lynxmotion, which has been modified to host a distributed sensory system. The complete control system was embedded on board and a wireless communication module was introduced for debugging purposes.

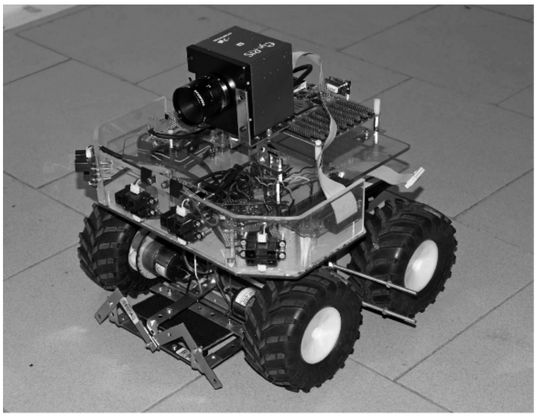


Fig. 6.5 Rover II: a dual drive roving robot endowed with a distributed sensory system

In the next section, more details about the robot and FPGA-based control architecture are given.

6.3.1 Spark Main Board

The FPGA-based development board preliminary used to assess the feasibility of the hardware control scheme showed some problems due to the big size of the board, the unused number of components, and mainly its lack of room within the FPGA to implement the needed algorithms. For these reasons, the design of a new specific board has been required.

Different approaches were taken into consideration during the design phase.

- One single board including a large FPGA (Stratix II EP2S180)
- One single board comprising two medium-size FPGA (Stratix II EP2S60)
- Two boards, each one with a medium-size EP2S60 FPGA connected via cables

After studying all the possibilities, the third option was adopted, as it is more robust in terms of developing time, costs, and flexibility. This solution permits us the use of only one board when the resources are enough for the application, and also two or even more boards when the algorithm requires more computational power. Figure 6.6 shows the SPARK 1.0 Main Board features in details. It represents a hardware platform for developing embedded systems based on Altera Stratix II EP2S60 FPGA with the following features:

1. Stratix II EP2S60F672C3N FPGA with more than 13,500 adaptive logic modules (ALM) and 1.3 million bits of on-chip memory
2. Eight expansion headers with access to 249 FPGA user I/O pins (3.3 V tolerant)
3. One expansion header with access to 16 FPGA user I/O pins (5 V tolerant)

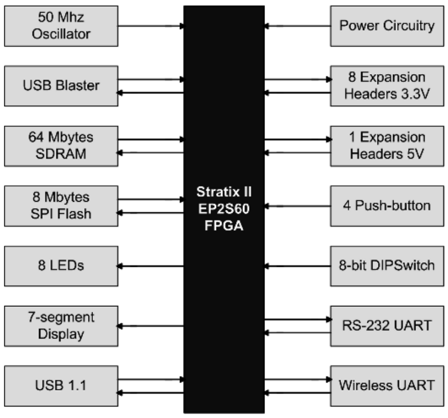


Fig. 6.6 SPARK board 1.0 block diagram

4. 1.2 V power supply
5. 5 V power supply
6. 12 V power supply connector
7. 3.3 V power supply
8. Eight LEDs connected to FPGA user I/O pins
9. Four push-button switches connected to FPGA user I/O pins
10. 32 Mbytes SDRAM
11. Dual seven-segment LED display
12. Integrated USB Blaster for connection with the ALTERA Software tools on a PC
13. 32 Mbytes of SDRAM
14. RS-232 serial connector with 5 V tolerant buffers
15. Wireless UART Communication Port
16. USB 1.1 Communication Port
17. 50 MHz clock signal for FPGA
18. 8 Mbytes SPI Flash Memory
19. Push-button switch to reboot the NIOS II processor configured in the FPGA
20. 8-bit DIP switch

A scheme with the component's locations on the SPARK board is shown in Fig. 6.7.

6.3.2 *Rover II*

The roving platform used for navigation experiments is a modified version of the dual drive Lynx Motion rover, called Rover II, whose dimensions are about $30 \times 30\,cm^2$. It is equipped with a bluetooth telemetry module, four infrared short

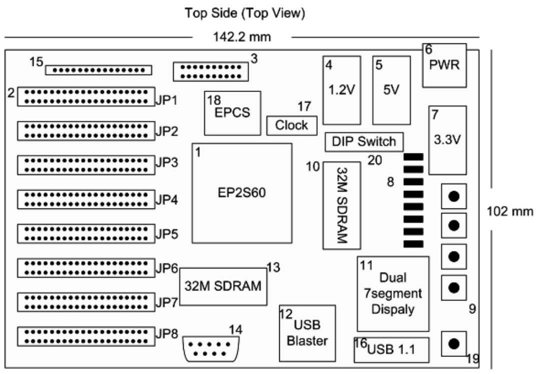


Fig. 6.7 Scheme of the SPARK board 1.0. A description of each block, numbered from 1 to 20, is reported in the text

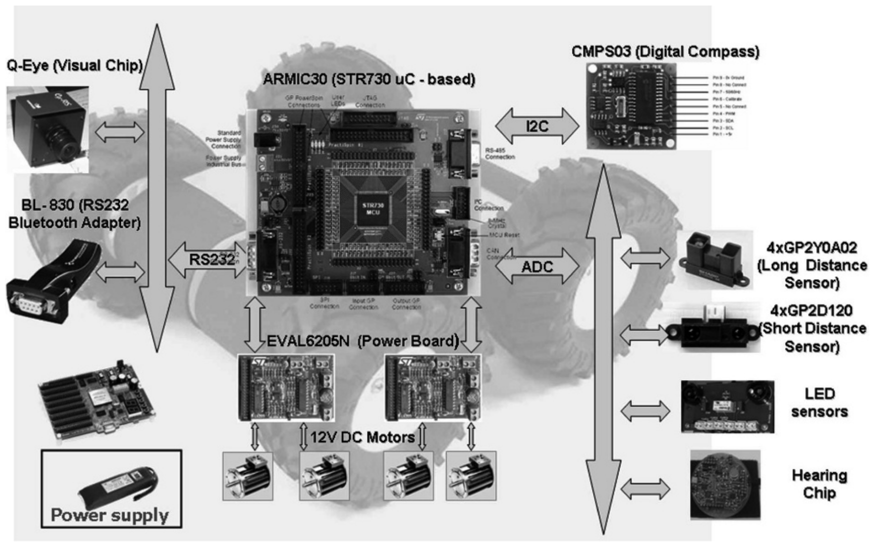

Q-Eye (Visual Chip)
ARMIC30 (STR730 uC - based)
CMPS03 (Digital Compass)
BL-830 (RS232 Bluetooth Adapter)
I2C
RS232
ADC
4xGP2Y0A02 (Long Distance Sensor)
4xGP2D120 (Short Distance Sensor)
EVAL6205N (Power Board)
LED sensors
12V DC Motors
Hearing Chip
Power supply

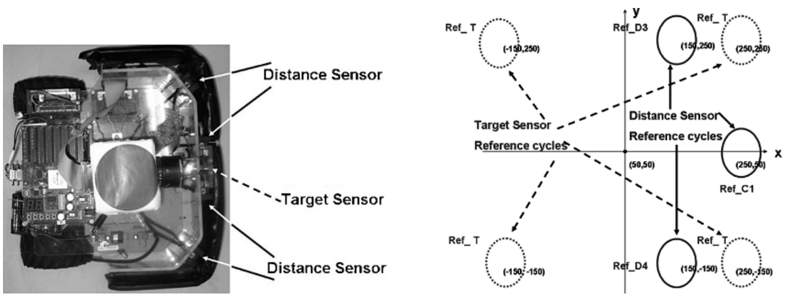
Fig. 6.8 Rover II control architecture

distance sensors Sharp GP2D120 (detection range 3–80 cm), four infrared long-distance sensors Sharp GP2Y0A02 (maximum detection distance about 150 cm), a digital compass, a low-level target detection system, a hearing board for cricket chirp recognition [34, 40] and with the Eye-RIS v1.2 visual system [3].

The complete control architecture, reported in Fig. 6.8, shows how the low-level control of the motors and the sensor handling is realized through a microcontroller STR730. This choice optimizes the motor performances of the robot, maintaining the high-level cognitive algorithms in the SPARK board and in the Eye-RIS visual system. Moreover, the Rover II can be easily interfaced with a PC through a bluetooth module, and this remote control configuration permits to perform some preliminary tests debugging the results directly on the PC.

In the following experiments, to fulfill a food retrieval task, distance sensors were used for obstacle avoidance, while a cricket-inspired hearing board was considered for the target detection issues [1].

Among the 5×5 possible reference cycles which can be used in the application reported here, only a subset of them has been effectively associated with a sensor. Their distribution in the phase plane *x*-*y* reflects the topological distribution of the robot sensors. Figure 6.9 shows a scheme of the robot equipped with four distance sensors and a target sensor. The corresponding reference cycles reported in the phase plane are related to the sensor positions. Distance sensors are directional, and each one is associated to a single reference cycle, while target sensor is characterized by an omnidirectional field of view, and for that reason it is associated to more than one (i.e., four) reference cycles [8].



(**a**) (**b**)

Fig. 6.9 Scheme of a Rover II robot (**a**) equipped with four distance sensors and a target sensor. In the phase plane $x - y$ (**b**), the reference cycles associated to each sensor are reported. Target sensor, due to its omnidirectional field of view, is associated to four reference cycles. Cycles can be generated by system (6.1) with following parameters: $a = b = c = d_1 = d_2 = 1$, $k_1 = k_2 = 50$, $h_1 = h_2 = 100$, $p_1 = p_2 = 1$, $q_1 = q_2 = 2$ and changing the offset. Equivalently, (6.7) can be used

The technique, based on placing reference cycles in the phase plane in accordance with the distribution of sensors on the robot, is important to directly connect the internal representation of the environment to the robot geometry. In our tests, only distance and hearing sensors have been used, although other sensors could be included, such us contact sensors, visual sensors, etc.

6.4 Hardware Implementation

In this section, the implementation of the framework for the control navigation algorithm presented in the previous section is discussed (see Fig. 6.10).

To realize the WCC approach for navigation control in hardware, the SPARK board was considered. Control algorithm was implemented on the Nios II soft-processor, and also customized VHDL (Very High Speed Integrated Circuits Hardware Description Language) blocks were used for the most time-consuming tasks.

In particular, the simulation and control of the multiscroll system are performed directly in a VHDL entity, named *sim_full*, while the NiosII microprocessor is devoted to handle the sensory system for the execution of the action selection layer and for the supervision of the activities of the VHDL entities implementing the WCC. When the simulation ends, the NiosII reads the parameters that identify the emerged cycle. Then it calculates the command to drive the roving robot. The simulation process implemented in the *sim_full* entity lasts for about 4.2 ms and the control algorithm running on NiosII for about 80 ms. Even if the RoverII robot could be driven via a bluetooth module, to reduce the communication time, the Spark board has to be mounted directly on the robot.

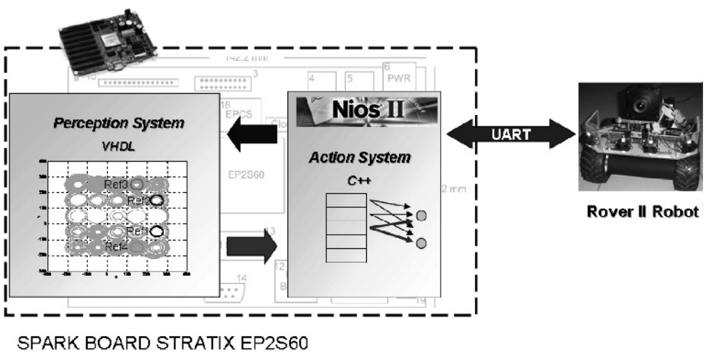


Fig. 6.10 Description of the framework used during the experiments

As discussed earlier, the control and simulation of the multiscroll system are made directly in VHDL entities. Other implementations were tested before testing the VHDL one. The first step was to simulate the chaotic system directly by using the NiosII in a C code algorithm. This solution works correctly, but the performances were not suitable for a real-time application. The high simulation time on NiosII (see Table 6.1) is probably due to the fact that its ALU (arithmetic logic unit) does not support natively floating point operations. This problem can be overcome by using custom instructions.

For these reasons, we went a step further; in fact, the "Soft-core" nature of the NiosII processor enables designers to integrate custom logic instructions (e.g., written in VHDL language) into the ALU. Similar to native NiosII instructions, custom instruction logic can take values from up to two source registers and optionally write back a result to a destination register. Also, in this case, the results were correct but the execution time, although decreased of one order of magnitude, remained too high for a real-time application. So, only the VHDL implementation is suitable as shown in Table 6.1.

Table 6.1 Performaces obtained with different implementations of the multiscroll control.

	Time[a]
Multiscroll control implemented on Nios II	10 s
Multiscroll control implemented on Nios II using custom instructions	1 s
Multiscroll control implemented in VHDL (*sim_full* entity)	4.2 ms

[a]The time is referred to the execution of 2,000 samples and the value of the integration step is 0.1

As the WCC is implemented directly in hardware, the role of the NiosII processors is limited to drive the activity of the block that implemented the *sim_full* entity to read the sensor values, to calculate the control gains, and to give motor commands to the robot on the basis of the actions generated by the action selection layer.

The main role of the *sim_full* entity consists in integrating the controlled third-order nonlinear multiscroll system for a given time. The integration algorithm implemented is a fourth order Runge–Kutta algorithm (RK4)[26]; the evolution time was fixed to 2,000 steps with an integration step of about 0.1. The evolution time length was experimentally chosen to guarantee the creation of "percepts" (i.e., low order dynamics like cycles) in the presence of sensorial stimuli. The output of the *sim_full* entity correspond to the behavior of the multiscroll system at the end of the integration process. These information are used by the Nios II to calculate robot actions.

For the implementation of the dynamical system, a decimal number arithmetic is needed. This algebra is not supported by the standard library of VHDL, and so the first step was to develop a dedicated library. As the fixed point operations are faster than the floating point ones, the fixed point arithmetic has been adopted. The total number of bits to codify the variable in the VHDL code is 24 bits. In particular, the most significant bit represents the sign: 14 bits are used to code the integer part and 10 bits code the fraction part.

In Fig. 6.11 the top level entity *sim_full* is shown. It is composed of two sub-entities:

- The *sim* block, which implements the simulation of the controlled multiscroll system adopting the RK4 algorithm with fixed integration step
- The *sim_machine*, which implements a finite state machine (FSM) that controls the activity of the *sim* entity.

The NiosII drives the *sim_full* entity through the input signals and provides the gains that are needed to control the simulated multiscroll system. Once the simulation is completed, it reads the results of the computation provided by the entity on its outputs.

Now we start with the description of the internal structure of the *sim* entity (Fig. 6.12). The *rk4*, *ram*, *structure_control_x*, and *structure_control_y* VHDL entities are the main components of this block. The function of the *rk4* entity is to implement the Runge–Kutta algorithm, and it needs the state vectors (x, y, z) and

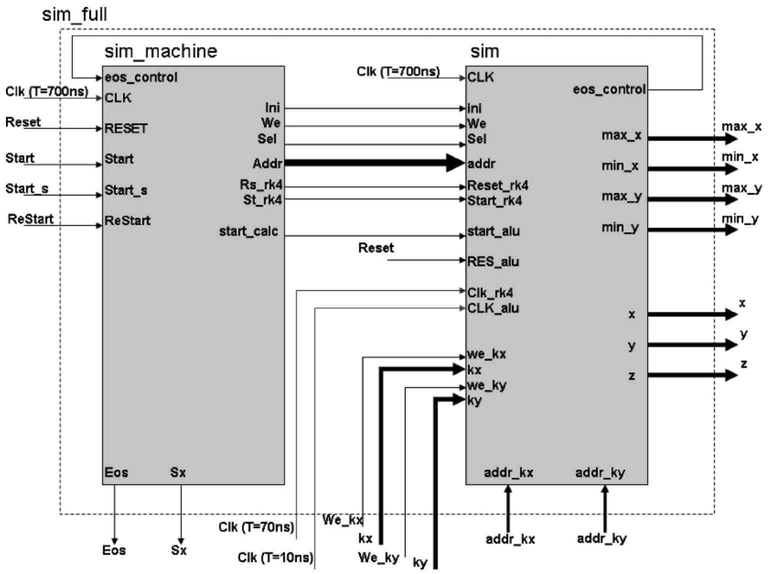


Fig. 6.11 VHDL Entity *sim_full*. The bold line represent a 24 bits length bus

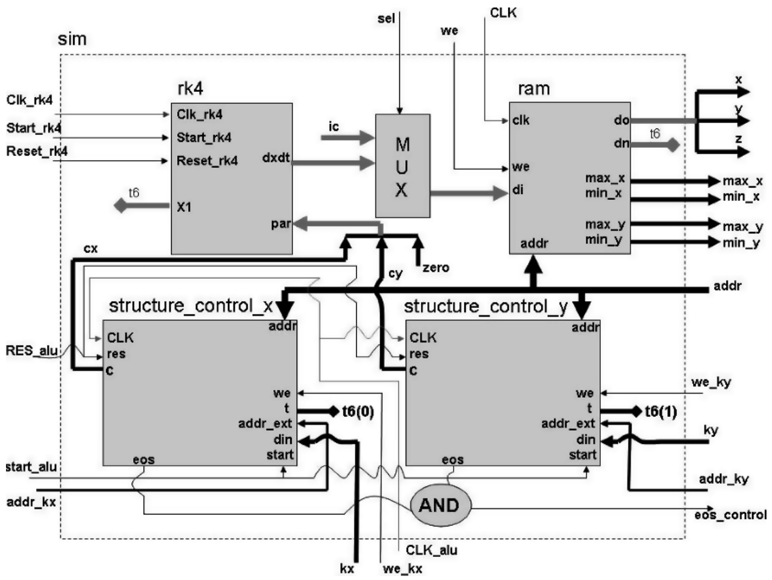


Fig. 6.12 VHDL Entity *sim*

the control signals to perform the integration. The function of the other inputs (i.e., *Clk_rk4*, *Start_rk4* and *Reset_rk4*) is to coordinate the activity of the entity.

The *ram* block is a memory that stores all the results of the integration: *di* is the data input, *do* is the data output, *dn* is the last sample produced and memorized, *we* is a write enable input, and *addr* is necessary to address the memory. The other outputs of this block represent the maximum and the minimum of the *x* and *y* state variables. This is an important feature that permits to determine the characteristics of the emergent cycle and to allow the use of the *sim_full* entity for the successive action selection steps.

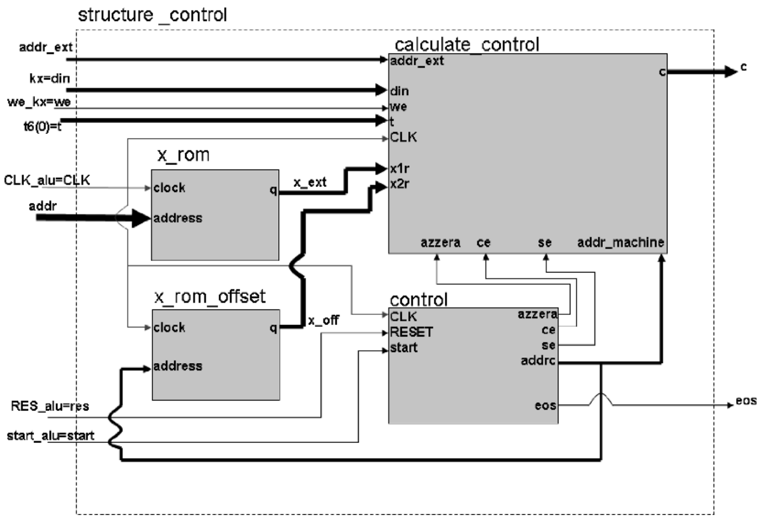


Fig. 6.13 VHDL Entity *structure_control*

The *structure_control_x* and *structure_control_y* blocks are two instances of the same VHDL entity named *structure_control*. The function of this entity is to calculate the control signals, while the k_{xi} and k_{yi} inputs represent the control gains corresponding to the reference *i* addressed with the input *addr_kx* and *addr_ky*, respectively. Figure 6.13 shows a block diagram of the entity for the *x* state variable. The reference signals are calculated off-line using (6.7), with x_{off} set to zero. The samples obtained are stored in the read-only memory (ROM) *x_rom*; the *x_rom_offset* ROM block stores the offset that must be added to the sample to obtain the desired reference signal; the *calculate_control* block computes the control gains and the *control* block is a finite state machine (FSM) that manages the activity of the entity.

Finally, in Fig. 6.14, the diagram of the FSM implemented is shown. The function of each state is reported in the following:

- *St_ini* sets the initial conditions, only at the startup
- *St_start* sets all the outputs to default values
- *St_sim* controls one step of simulation
- *St_mem* stores the produced samples into the *ram* block
- *St_eos* notifies the end of simulation
- *St_load, St_send, St_inc* together with the *Start_s* input and the *sx* output allow the NiosII processor to access correctly all the simulation parameters in memory.

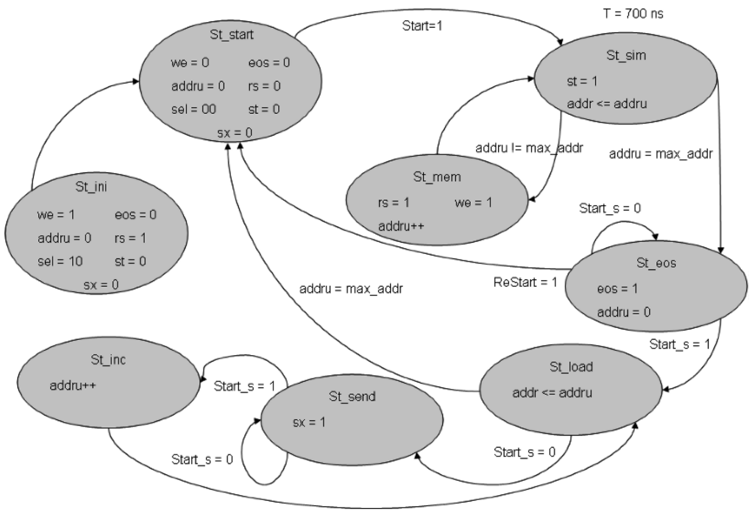


Fig. 6.14 VHDL Entity *sim_machine*

As this process is not needed (due to the fact that the parameters needed to identify the cycle are available in the *St_eos* state), the *ReStart* permits to skip it, and so to optimize the performances.

The inputs *Clk*, *Start*, *Reset* allow the control of the FSM activity.

Figure 6.15 represents an example of the evolution of the controlled multiscroll system implemented in FPGA. The hardware results are equivalent to the simulation ones, and so the fixed point algebra and the *sim_full* VHDL entity are suitable for implementing the WCC.

Table 6.2 SPARK board FPGA resources needed to implement the WCC.

Resource	Used
ALUTs	27 %
Memory	34 %
DSP	56 %

Finally, Table 6.2 shows the FPGA resources of the SPARK board used to implement the proposed control architecture.

6.5 Experiments

The proposed action-oriented perceptual architecture was tested both in a simulated environment and in a real arena. The simulation tool used to evaluate the system capabilities is called SPAN [8], and has been already used to compare the reactive

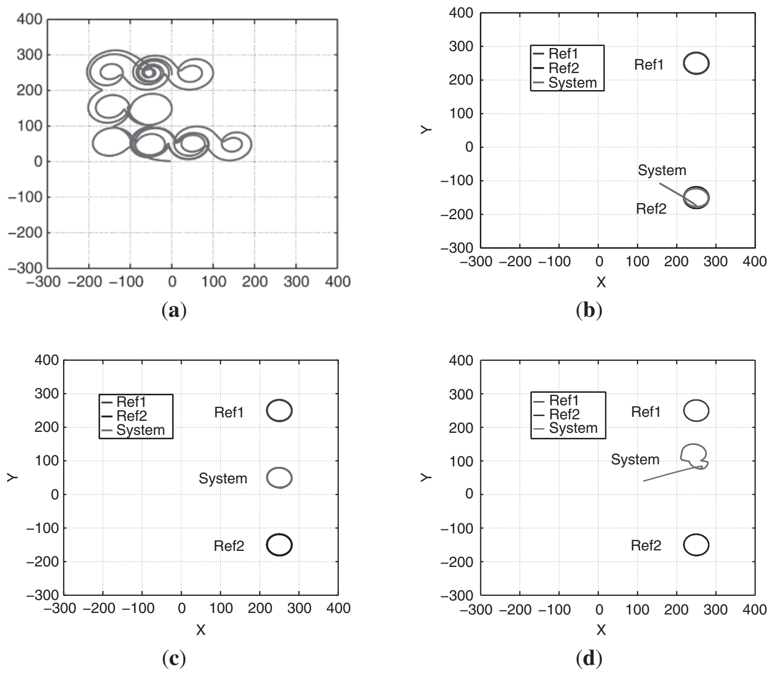


Fig. 6.15 An example of the evolution of the multiscroll system when controlled by reference dynamics associated with sensors. (**a**) When there are no input sensorial stimuli, the system shows a chaotic behavior; (**b**) when a single stimulus is perceived, the system converges to the reference cycle; (**c**),(**d**) when a different stimuli is perceived, the resulting cycle is placed between the references and the position depends on the gain value associated to the references

layer of the WCC approach (i.e., with a deterministic association between internal dynamics and corresponding robot actions) with more traditional approaches to navigation like the potential field (see [6] for more details). Here we want to demonstrate how a robot can learn to associate rewarding behaviors to the different environmental situations encountered, aiming at performing a given task that in the following experiments consists in retrieving targets avoiding obstacles.

6.5.1 Experimental Setup

During the learning phase, a growing set of emerged cycles, arisen in response to different environmental conditions, are associated to suitable actions through the MM algorithm. To evaluate the robot's performances in a quantitative way, the following parameters have been considered:

- P_{new}: cumulative number of new perceptual states that emerge during learning
- *Bumps*: cumulative number of collisions with obstacles

- *Explored Area*: new area covered (i.e., exploration capability)
- *Retrieved Targets*: number of targets retrieved in the environment

Initially each new pattern is associated with a random action, but the continuous emerging of such a pattern leads the action selection network to tune its weights to optimize the association between the perceptual state and the action to be performed. It is also desirable that new patterns occur only during the first learning steps (i.e., epochs). To guarantee the convergence of the algorithm, the learning process cannot be considered as ended while new patterns continue to emerge with a high frequency. Moreover, to solve the robot navigation problem, it is necessary that a pattern occurs several times, as the robot learns by trial and error.

As the term a_{s_q} gives information about the stability of the action associated with the pattern q, we use this parameter to evaluate the convergence of the learning process. The LRU algorithm (that manages the pattern table) was modified to consider a_{s_q}. The pattern q cannot be replaced if its a_{s_q} is under a fixed threshold (AS_{Learn}) that is determined during the design phase.

The code that identifies an emerged cycle is constituted by the four parameters x_q, y_q, $\bar{x}_q$, and $\bar{y}_q$ that assume continuous values, because they depend on the evolution of the state variables of the controlled system.

The choice of the tolerance to distinguish among different patterns is a crucial problem during the design phase. If the tolerance increases, the number of patterns representing the robot's perception of the environment decreases. Then, the learning time is reduced but the perception–action association is more rough. On the contrary, if the tolerance is reduced, the number of actions increases, producing a wider range of different solutions for the navigation task. In this way it is feasible to reach a better solution, but it is more time consuming. Table 6.3 shows the value of the most relevant parameters used during the learning phase. The target sensor equipped on the robot, based on cricket's phonotaxis, is not able to estimate accurately the distance between robot and sound source, for this reason the parameter h_D used in (6.11) has been not considered.

6.5.2 Experimental Results

The environment used to perform tests and comparisons is a $10 \times 10\,\text{m}^2$ room with three obstacles and two targets. The simulated arena and the real arena are shown in Fig. 6.16. Both in simulation and in the real case, a set of five learning trials was performed, with the MM structure randomly initialized. The learning phase is maintained for 1,200 epochs (i.e., actions) that correspond to 40 min for the real robot and a few minutes in the virtual arena. Each robot simulation step (i.e., epoch) corresponds to a single robot action: this is determined simulating the dynamical system for 2,000 steps with an integration step equal to 0.1. These parameters guarantee the convergence of the multiscroll system to a stable attractor (i.e., a cycle) when external stimuli are perceived by the robot; otherwise a chaotic evolution is shown.

Table 6.3 Relevant parameters of the MM-like structure.

RF parameters		Learning parameters	
k_i for frontal distance sensors	15	a_s start value	0.6
k_i for lateral distance sensors	10	a_s decrement factor	0.01
h_A	10	Tolerance	8%

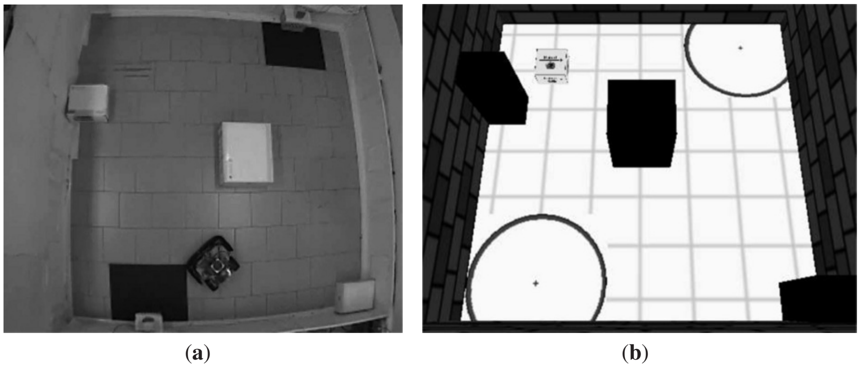
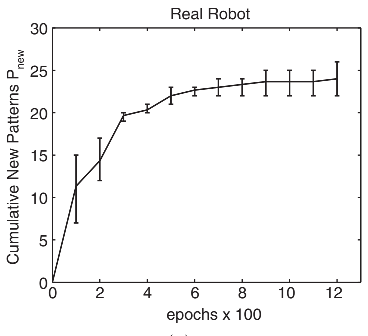


Fig. 6.16 3D view of the environment used during the learning phase in real (**a**) and simulated experiments (**b**). The dimensions of the environments are $10 \times 10\,\mathrm{m}^2$

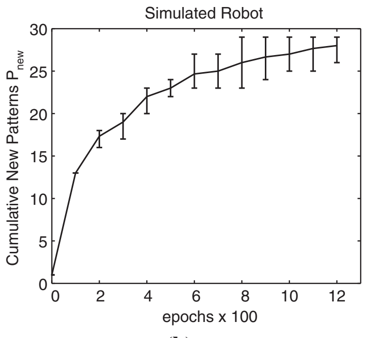
During the learning phase, a sequence of new patterns, in the form of cycles, emerges and the robot learns how to behave in the current situation. To evaluate the convergence of the learning phase in Fig. 6.17, the trends of the cumulative number of new patterns that arise (P_{new}) is shown in the case of the simulated agent and the roving robot. The learning process leads to a huge improvement of the robot behavior for the situation (i.e., perceptual state) that more often occurs, while some other patterns cannot be suitably learned if they seldom emerge.

After the first learning epoches, the number of new emerging patterns is very low and a steady state condition is reached after 100 epoches. These results were obtained adopting the learning parameters defined in Table 6.3. In particular, the total number of patterns that emerges, around 25–30, is directly related to the tolerance parameter that has been set to 8%. In other simulations carried out, increasing the tolerance factor to 15% reduces the number of emerging patterns to 20%. A reduced number of emerged patterns leads to the speed-up of the learning phase, but decreases the specialization of the robot actions with a consequent reduction of the performances.

a_s is an important parameter to evaluate the efficacy of the learning phase. The value of a_s is directly related to the stability of the association among perceptual state and robot action. The total number of emerged patterns is about 25–30 and

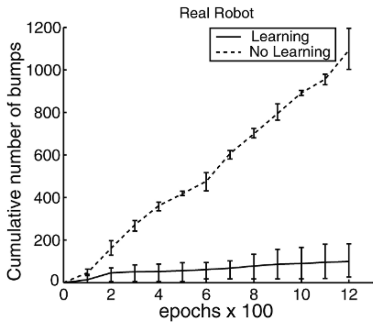


(**a**)

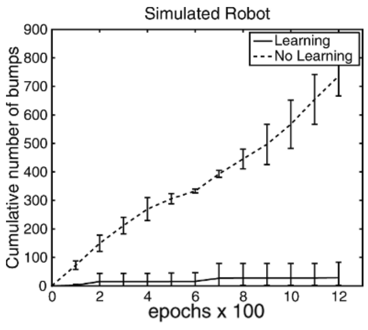


(**b**)

Fig. 6.17 Cumulative number of new patterns that emerge during the learning phase calculated in windows of 100 epochs in the real environment (**a**) and in the simulated arena (**b**). The *bars* indicate the minimum and maximum value while the *solid line* is the mean value of the set of five trials performed



(**a**)



(**b**)

Fig. 6.18 Cumulative value of bumps calculated in windows of 100 epochs in the conditions of learning and no-learning for the real (**a**) and simulated (**b**) environments. The two trends span among the minimum and maximum value for each window

more than 60% of the patterns have an $a_s < 0.5$: this corresponds to more than 100 updates of the associated action following the indication of the reward function, and about the 25% of the patterns have an $a_s < 0.1$ that correspond to more than 500 updates.

The learning process guided by the reward function significantly improves the robot capabilities evaluated in terms of the number of bumps and target retrieved. In Figs. 6.18 and 6.19, the cumulative number of bumps and targets found is shown for the two learning cases, comparing the behavior of the system during learning with the same architecture when the learning is not activated. The results show that a significant difference in terms of performance is evident.

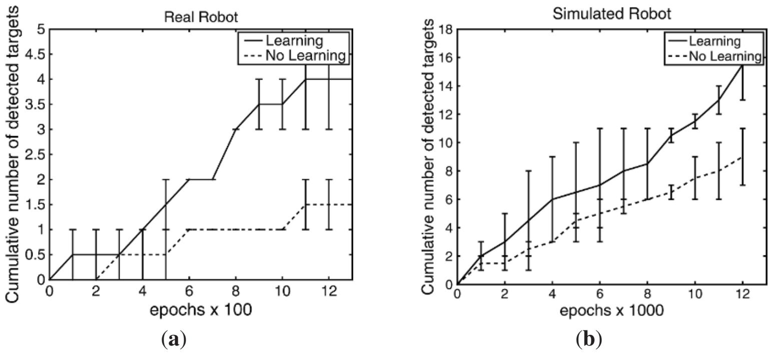


Fig. 6.19 Cumulative value of retrieved targets calculated in windows of 100 and 1,000 epochs in the conditions of learning and no-learning for the real (**a**) and simulated (**b**) environments. The two trends span among the minimum and maximum value for each window

As far as the avoidance behavior is concerned, the robot learns how to manage the information coming from distance sensors to avoid bumps. The improvement is evident both in simulation and in the real experiments.

Another important index to be considered is the number of targets retrieved. When a target is found by the robot, it is turned off until another target is reached. Because of the limited power supply, the robot experiment was limited to 1,200 epoches, while the simulation was extended to 12,000 epoches to better appreciate the improvement over a more relevant number of target retrieving (see Fig. 6.19). The number of target found by the real robot when the learning process does not work is very low, in fact the robot is often trapped by the obstacles placed in the arena. The WCC performances were also compared with other more traditional navigation algorithms showing similar results [6].

To understand how the robot behavior is modified during the learning process, in Fig. 6.20, the trajectories followed by the robot before and after learning are shown.

Cmparison between the performance with and without learning outlines the capability of the control system to create a suitable link between perception and action.

To further outline the results of the testing phase for a learned architecture, information on the association between perceptual patterns and corresponding actions is reported. In particular, in Fig. 6.21, the final emerged actions associated to the mostly used patterns are shown. Figure 6.21 shows the x-y phase plane of the multiscroll system together with the internal patterns emerged during a learning phase. For sake of clarity, each class is reported only with a marker, indicating its position (i.e., parameters x_q and y_q). The vector associated to each pattern shows module and phase of the corresponding action performed by the robot with respect to the x-axis that indicates frontal direction of the robot motion.

It is important to notice that, in this case, the robot through learning is able to define a series of actions needed in specific situations that can be adopted to accomplish the food retrieval task. The approach can be extended to other tasks,

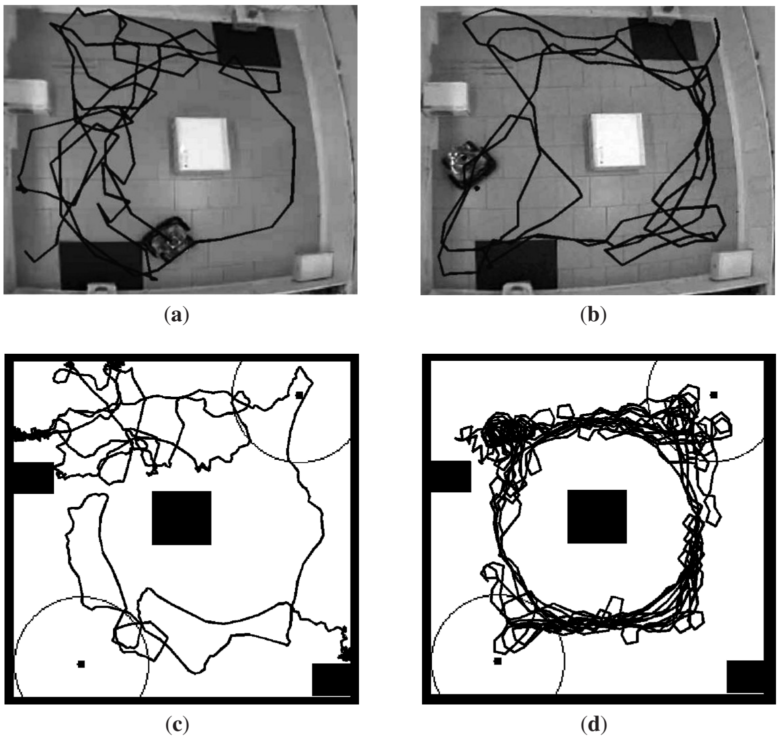


Fig. 6.20 Trajectories followed by the robot during a test in the learning environments. (**a**) and (**b**) Behavior of the robot without learning; (**c**) and (**d**) Trajectories followed after the learning phase

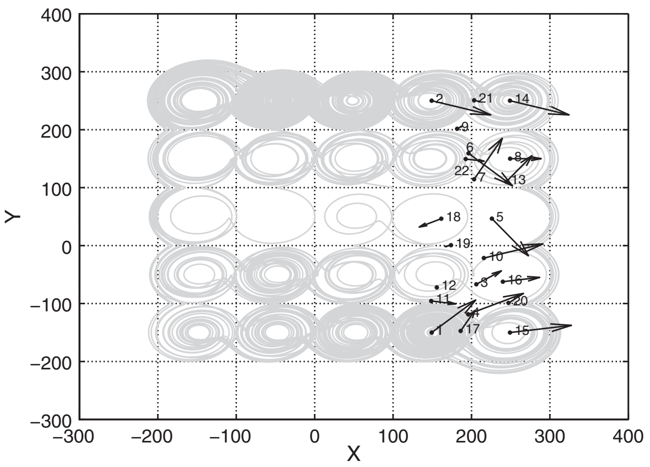


Fig. 6.21 *Action Map* created during the learning phase by the Rover II. Each vector indicates the phase action associated with the emerged pattern indicated only with its position in the phase plane (x_q, y_q)

considering other sensors and also different time scales, associating to the robot's internal state of not only a simple action but also a complete behavior.

Further experiments together with multimedia materials are available on the web [37].

6.6 Summary and Remarks

The introduced strategy is based on the WCC technique, with the addition of a plastically adaptable self learning layer. The technique is based on a state feedback approach and the control gains were here chosen to grant the controlled system stability under stimulation with reference signals (see [6] for details). The term *WCC* refers to a strategy that does not aim to the exact matching between the reference and the controlled signal; instead, the chaotic signal has to collapse to an orbit near the reference signal. At this point, the amplitude value of the control gains is related to the matching degree between the reference and the controlled signal. The control gain value could be therefore used as an additional choice to weight a kind of "degree of attention" that the learning system could pay to the corresponding sensor signal. If learning is used also to choose the control gains, at the end of the learning phase, the robot could be allowed to discard useless sensor signals and to "pay attention" to the important ones. This approach is currently under active investigation. Moreover, the strategy introduced in this chapter was applied to the apparently simple task of autonomous navigation learning. The clear advantages over the classical approaches, for example, related to the potential field, are that the control structure, based on the multiscroll system, is quite general. The fact that the results obtained are comparable with those of the potential field is relevant (see [6] for details). This means that a general approach to learning the sensing–perception–action cycle using the power of information embedding typical of chaotic systems, applied to a traditional task, succeeds in reaching the same results as a technique peculiarly designed to solve that task. The WCC-MM approach can in fact be applied to learn an arbitrary *action-map* or, in general, a *behavior map*. In particular, we exploit the rich information embedding capability of a chaotic system with a simple learning that gives a "meaning" to the embedded information (taking inspiration from [19], within the context of the robot action. This contextualization is decided through the reward function definition. According to the best of the authors' knowledge, it is the first time that the chaotic circuits and system theory, linked to a simple neural learning, is used to approach problems relevant in robot perception, even in the simple case of navigation. In fact, here navigation is treated as a perceptual task. Several other examples are currently under investigation to further generalize the approach and a considerable theoretical effort is being nowadays paid to include the strategy introduced here within a more general scheme for robot perception in unknown conditions and environments.

6.7 Conclusion

In this chapter, a new architecture used to fulfill the sensing–perception–action loop has been proposed. To evaluate the main characteristics of the proposed approach, a case study has been considered: navigation control of a roving robot in unknown environments. The method based on the control of a chaotic multiscroll system allows to synthesize a perceptual scheme in a compact form easy to be processed. Moreover, an action layer has been introduced to associate with the robot's internal states, new behaviors through an unsupervised learning driven by a reward function. The architecture has been implemented in an FPGA-based hardware designed to fulfill the flexibility needed by a cognitive architecture. A completely autonomous roving robot, equipped with a suite of sensors used to perform a food retrieval task, was tested, showing the improvement, in terms of robot capabilities, obtained through a reward-based learning.

As already envisaged, the chaotic dynamical system representing the core of the perceptual layer allows to extend the sensory system in a very simple and modular way. Further developments will include the introduction in the real robot of other kinds of sensors like visual, needed to improve the system capabilities to accomplish the assigned task.

References

1. Alba, L., Arena, P., De Fiore, S., Listan, J., Patané, L., Scordino, G., Webb, B.: Multi-sensory architectures for action-oriented perception. In: Proceedings of Microtechnologies for the New Millennium (SPIE'2007). Gran Canaria, Spain (2007)
2. Altera Corporation: Altera home page. `http://www.altera.com`.
3. Anafocus sl: Eye-ris producer home page. `http://www.anafocus.com`.
4. Aradi, I., Barna, G., Erdi, P.: Chaos and learning in the olfactory bulb. Int. J. Intell. Syst. **10**(1), 89–117 (1995)
5. Arena, P., Crucitti, P., Fortuna, L., Frasca, M., Lombardo, D., Patané, L.: Turing patterns in rd-cnns for the emergence of perceptual states in roving robots. Bifurcation Chaos **17**(1), 107–127 (2007)
6. Arena, P., De Fiore, S., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: Reactive navigation through multiscroll systems: from theory to real-time implementation. Autonomous Robots **25**(1–2), 123–146 (2007)
7. Arena, P., Fortuna, L., Frasca, M., Hulub, M.: Implementation and synchronization of 3×3 grid scroll chaotic circuits with analog programmable devices. Chaos **16**(1) (2006)
8. Arena, P., Fortuna, L., Frasca, M., Lo Turco, G., Patané, L., Russo, R.: A new simulation tool for actionoriented perception systems. In: Proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation (EFTA'2005), pp. 19–22. Catania, Italy (2005)
9. Arena, P., Fortuna, L., Frasca, M., Lombardo, D., Patané, L.: Learning efference in cnns for perception-based navigation control. In: Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA'2005), pp. 18–21. Bruges, Belgium (2005)
10. Arena, P., Fortuna, L., Frasca, M., Pasqualino, R., Patané, L.: Cnns and motor maps for bio-inspired collision avoidance in roving robots. In: 8th IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA'2004). Budapest (2004)

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. Arena, P., Fortuna, L., Frasca, M., Patané, L., Pavone, M.: Towards autonomous adaptive behavior in a bio-inspired cnn-controlled robot. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'2006), pp. 21–24. Island of Kos, Grecia (2006)
13. Arena, P., Patané, L.: Spatial Temporal Patterns for Action Oriented Perception in Roving Robots. Springer, Berlin (2008)
14. Arkin, R.C.: Behavior-Based Robotics. MIT Press, Cambridge, MA (1998)
15. Boccaletti, S., Grebogi, C., Lai, Y.C., Mancini, H., Maza, D.: The control of chaos: theory and applications. Phys. Rep. **329**, 103–197 (2000)
16. Freeman, W.J.: The physiology of perception. Sci. Am. **264**(2), 78–85 (1991)
17. Freeman, W.J.: Characteristics of the synchronization of brain activity imposed by finite conduction velocities of axons. Bifurcation Chaos **10**(10), 2307–2322 (1999)
18. Freeman, W.J.: A neurobiological theory of meaning in perception. Part I: Information and meaning in nonconvergent and nonlocal brain dynamincs. Bifurcation Chaos **13**(9), 2493–2511 (2003)
19. Freeman, W.J.: How and why brains create meaning from sensory information. Bifurcation Chaos **14**(2), 515–530 (2004)
20. Freeman, W.J., Kozma, R.: Local-global interactions and the role of mesoscopic (intermediate-range) elements in brain dynamics. Behav. Brain Sci. **23**(3), 401 (2000)
21. Gutierrez-Osuna, R., Gutierrez-Galvez, A.: Habituation in the kiii olfactory model with chemical sensor arrays. IEEE Trans. Neural Netw. **14**(6), 1565–1568 (2003)
22. Harter, D., Kozma, R.: Chaotic neurodynamics for autonomous agents. IEEE Trans. Neural Netw. **16**(3), 565–579 (2005)
23. Kohonen, T.: Self-organized formation of topologically correct feature maps. Bio. Cybern. **43**(1), 59–69 (1972)
24. Kozma, R., Freeman, W.J.: Chaotic resonance – methods and applications for robust classification of noisy and variable patterns. Bifurcation Chaos **11**(6), 1607–1629 (2000)
25. Kuniyoshi, Y., Sangawa, S.: Emergence and development of motor behavior from neural-body coupling. In: Proc. IEEE Int. Conference on Robotics and Automation (ICRA'2007). Rome, Italy (2007)
26. Lapidus, L., Seinfeld, J.: Numerical Solution of Ordinary Differential Equations. Academic Press, New York (1971)
27. Lu, J., Chen, G., Yu, X., Leung, H.: Design and analysis of multiscroll chaotic attractors from saturated function series. IEEE T Circuits-I **51**(12), 2476–2490 (2004)
28. Makarov, V.A., Castellanos, N.P., Velarde, M.G.: Simple agents benefits only from simple brains. Trans. Eng. Comput. Tech. **15**, 25–30 (2006)
29. Manganaro, G., Arena, P., Fortuna, L.: Cellular Neural Networks: Chaos, Complexity and VLSI processing. Springer, Berlin (1999)
30. Murray, J.D.: Mathematical Biology. Springer, Berlin (1993)
31. Ott, E., Grebogi, C., Yorke, J.A.: Controlling chaos. Phys. Rev. Lett. **64**(11) (1990)
32. Pyragas, K.: Continuos control of chaos by self-controlling feedback. Phys. Lett. A **170**, 421–428 (1992)
33. Pyragas, K.: Predictable chaos in slightly pertirbed unpredictable chaotic systems. Phys. Lett. A **181**, 203–210 (1993)
34. Reeve, R., Webb, B.: New neural circuits for robot phonotaxis. Phil. Trans. R. Soc. A **361**, 2245–2266 (2002)
35. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-Organizing Maps. Addison Wesley, Reading, MA (1992)
36. Skarda, C.A., Freeman, W.J.: How brains make chaos in order to make sense of the world. Behav. Brain Sci. **10**, 161–195 (1987)
37. SPARK Project: Spark eu project home page. `http://www.spark.diees.unict.it`
38. Steels, L., Brooks, R.A.: The Artificial Life Route to Artificial Intelligence: Building Embodied Situated Agents. Lawrence Erlbaum Associates, Hillsdale (1995)

39. Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents. IEEE T. Evolut. Comput. **11**(2), 151–180 (2007)
40. Webb, B., Scutt, T.: A simple latency dependent spiking neuron model of cricket phonotaxis. Biol. Cybern. **82(3)**, 247–269 (2000)
41. Yalcin, M.E., Suykens, J.A.K., Vandewalle, J., Ozoguz, S.: Families of scroll grid attractors. Bifurcation Chaos **12**(1), 23–41 (2002)