Chapter 6
From low to high level approach to cognitive control

P. Arena, S. De Fiore, M. Frasca, D. Lombardo and L. Patané

Abstract In this Chapter the application of dynamical systems to model reactive and precognitive behaviours is discussed. We present an approach to navigation based on the control of a chaotic system that is enslaved, on the basis of sensory stimuli, into low order dynamics that are used as percepts of the environmental situations. Another aspect taken into consideration, is the introduction of correlation mechanisms, important for the emergence of anticipation. In this case a spiking network is used to control a simulated robot learning to anticipate sensory events. Finally the proposed approach has been applied to solve a landmark navigation problem.

6.1 Introduction

In living beings, cognitive capabilities are based on inherited behaviours that are important for their survival. These reactive behaviours, triggered by external stimuli, are the basic blocks of a cognitive architecture.

In this Chapter we propose a new technique, called weak chaos control technique, that has been used to implement the reactive layer of a sensing-perception-action scheme. This control mechanisms is functionally inspired by the research activity of Prof W. Freeman and coauthors, on the formation of percepts in the olfactory bulb in rabbits. The proposed model has been designed to be embedded in hardware for real-time control of roving robots.

The definition of a strategy for the generation of reactive behaviour represents a first step toward the realization of a cognitive control. Growing up in complexity, the importance of a correlation layer is evident. To discuss on these aspects, we introduce a network of spiking neurons devoted to navigation control. Three different examples, dealing with stimuli of increasing complexity are investigated. First, a

P. Arena, S. De Fiore, M. Frasca, D. Lombardo and L. Patané
Department of Electrical, Electronic and System Engineering, University of Catania, I-95125 Catania, Italy, e-mail: [parena, lpatane]@diees.unict.it

simulated robot is controlled to avoid obstacles through a network of spiking neurons. Then, a second layer is designed aiming to provide the robot with a target approach system, which makes the robot able to move towards visual targets. Finally, a network of spiking neurons for navigation based on visual cues is introduced. In all the cases it has been assumed that the robot knows some a priori responses to low level sensors (i.e. to contact sensors in the case of obstacles, to proximity target sensors in the case of visual targets, or to the visual target for navigation with visual cues) and has to learn the response to high level stimuli (i.e. range finder sensors or visual input). The biologically plausible paradigm of Spike-Timing-Dependent-Plasticity (STDP), already introduced in Chapter 3 is included in the network to make the system able to learn high level responses that guide navigation through a simple unstructured environment. The learning procedure is based on classical conditioning.

To conclude the Chapter, a new methodology for landmark navigation, based on correlation mechanisms, is introduced. Either for animals or for artificial agents, the whole problem of landmark navigation can be divided into two parts: first, the agent has to recognize, from the dynamic environment, space invariant objects which can be considered as suitable "landmarks" for driving the motion towards a goal position; second, it has to use the information on the landmarks to effectively navigate within the environment. Here, the problem of determining landmarks has been addressed by processing the external information through a spiking network with dynamic synapses plastically tuned by an STDP algorithm. The learning processes establish correlations between the incoming stimuli, allowing the system to extract from the scenario important features which can play the role of landmarks. Once established the landmarks, the agent acquires geometric relationships between them and the goal position. This process defines the parameters of a recurrent neural network (RNN). This network drives the agent navigation, filtering the information about landmarks given within an absolute reference system (e.g the North). When the absolute reference is not available, a safety mechanism acts to control the motion maintaining a correct heading. Simulation results showed the potentiality of the proposed architecture: this is able to drive an agent towards the desired position in presence of stimuli subject to noise and also in the case of partially obscured landmarks.

6.2 Weak Chaos Control for the generation of reflexive behaviours

All forms of adaptive behavior require the processing of multiply sensory information and their transformation into series of goal-directed actions. In the most primitive animal species the entire process is regulated by external (environmental) and internal feedback through the animal body [51, 26].

Cortical processes information coming from objects identified in the environment through spike trains from receptors by enrolling dedicated neural assemblies. These

are nonlinear dynamical coupled systems whose collective dynamics constitutes the mental representation of the stimuli. Freeman and co-workers, in their experimental studies on the dynamics of sensory processing in animals [24, 25], conceived a "dynamical theory of perception". The hypothesis is that cerebral activity can be represented by a chaotic dynamics. They attained this result by different experiments on rabbits which inhaled in a pre-programmed way several smells. Through the electroencephalogram (EEG), Freeman evaluated the action potentials in the olfactory bulb and he noticed that the potential waves showed a complex behavior. So he came to the conclusion that an internal mental representation (cerebral pattern) of a stimulus is the result of a complex dynamics in the sensory cortex in cooperation with the limbic system that implements the supporting processes of intention and attention [22]. More in details, according to Freeman [25], the dynamics of the olfactory bulb is characterized by a high-dimensional chaotic attractor with multiple wings. The wings can be considered as potential memory traces formed by learning through the animal's life history. In the absence of sensory stimuli, the system is in a high dimensional itinerant search mode, visiting various "wings". In response to a given stimulus, the dynamics of the system is constrained to oscillations in one of the wings, which is identified with the stimulus. Once the input is removed, the system switches back to the high-dimensional, itinerant basal mode.

Analyzing the experimental data acquired, Freeman proposed a dynamical model of the olfactory system, called K-sets [21, 47]. The model is able to show all the chaotic and oscillatory behaviours identified through the experiments. Freeman and Skarda [47] discussed on the important role of chaos in the formation of perceptual meanings. Accordingly to their works, neural system activity persists in a chaotic state until sensors perturb this behaviour. The result of this process is that a new attractor emerges representing the meaning of the incoming stimuli. The role of chaos is fundamental to provide the flexibility and the robustness needed by the system during the migration through different perceptual states. A discrete implementation of Freeman's K model (i.e. KA sets) was developed and applied to navigation control of autonomous agents [29]. The controller parameters have been learned through an evolutionary approach [28] and also by using unsupervised learning strategies [29]. Our main objective here is to propose a reactive control architecture for autonomous robots, taking care of the functional properties discovered by Freeman in the olfactory bulb. The idea is to use a simple but chaotic dynamical system with suitable characteristics that can functionally simulate the creation of perceptual patterns. The patterns can be used to guide the robot actions and the control system can be easily extended to include a wide number of sensors. Furthermore the control architecture can be implemented at a hardware level in an FPGA-based board to be embedded on an autonomous roving robot [4].

The perception stage is represented by a suitable chaotic attractor, controlled by incoming signals from sensors. In particular a multidimensional state feedback control strategy has been implemented. A peculiarity of the approach is that, whereas most of the chaos control techniques are focused on the control of the chaotic trajectories towards equilibrium points or native limit cycles, in this case the controlled system is able to converge also to orbits never shown by the uncontrolled system.

This is due both to the particular system chosen and to the fact that a multireference control is required. For this reason, we called this technique "weak chaos control" (WCC). The crucial advantages of this approach are the compact representation of the perception system, the real time implementation in view of its application to robot navigation control and the possibility to take into consideration the physical structure of the robot within the environment where it moves. This characteristic is realised since the robot geometry is introduced within the phase space, where chaotic wanderings are represented. Moreover obstacles or target positions are mapped in the phase space directly reflecting their actual position, with respect to the robot, as in the real environment. We have therefore, in the phase space, a kind of mirrored real environment, as measured by the sensors. The emerging controlled orbit will influence the behaviour of the robot by means of suitable actions, gained through the implementation of a simple unsupervised learning phase, that has already been presented in [3].

6.2.1 The chaotic Multiscroll system

In this section the chaotic circuit used as perceptual system is introduced. Since this system should be able to deal with a great number of sensorial stimuli and represent them, a chaotic system, able to generate multiscrolls [38], has been adopted. This 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 a chaotic attractor consisting of multiply scroll distributed in the phase space (i.e. *n*-scrolls attractor) [39]. It is able to generate one-dimensional (1-D) *n*-scrolls, two-dimensional (2-D) $n \times m$-grid scrolls or three-dimensional (3-D) $n \times m \times l$-grid scroll chaotic attractors by using saturated function series. In this work a 2-D multiscroll system has been chosen. It is described by the following differential equations [38]:

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

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

$$g_i(x;k_j;h_j) = \begin{cases} 2k_j & if\, x > ih_j + 1, \\ k_j(x - ih_j) + k_j & if\, |x - ih_j| \leq 1, \\ 0 & if\, x < ih_j - 1 \end{cases} \tag{6.3}$$

$$g_{-i}(x;k_j;h_j) = \begin{cases} 0 & if\, x > -ih_j + 1, \\ k_j(x + ih_j) - k_j & if\, |x + ih_j| \leq 1, \\ -2k_j & if\, x < -ih_j - 1 \end{cases} \tag{6.4}$$

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 [38] to generate a 2-D 5×5 grid of scroll attractors. An example of the chaotic dynamics of system (6.1) is given in Fig. 6.1.

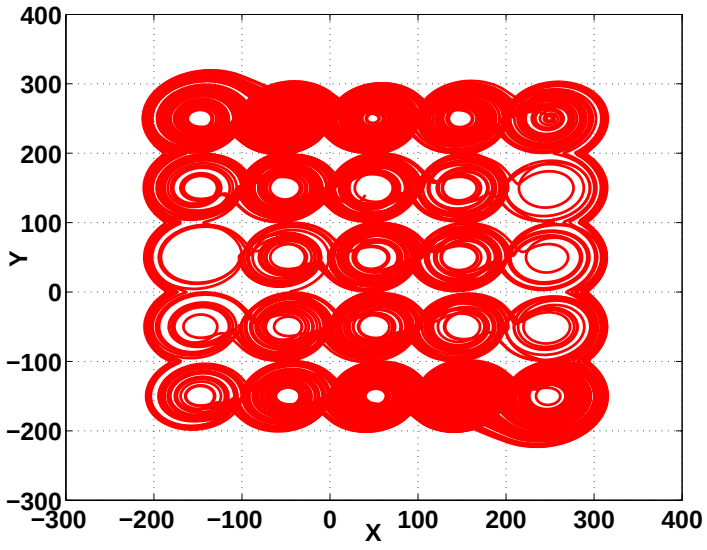


Fig. 6.1 Projection of the 5x5 grid of *scroll attractors* in the plane *x*-*y*.

6.2.2 Control of the multiscroll system

In our approach the perceptual system is represented by the multiscroll attractor of equations (6.1), whereas sensorial stimuli can interact with the system through (constant or periodic) inputs that can modify the internal chaotic behavior. Since 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.

Briefly, chaos control refers to a process wherein a tiny perturbation is applied to a chaotic system in order to realize a desirable behavior (e.g. chaotic, periodic and others). Several techniques have been developed for the control of chaos [11].

In view of our application, a continuous-time technique like the Pyragas's method is a suitable choice [42].

In this method [42, 43] the following model is taken into account:

$$\frac{dy}{dt} = P(y,x) + F(t), \qquad \frac{dx}{dt} = Q(y,x) \tag{6.5}$$

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 which forces the chaotic system to follow the desired dynamics. Pyragas [42, 43] 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:

$$F(t) = K[\widehat{y}(t) - y(t)] \tag{6.6}$$

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.

6.2.2.1 Control scheme

In our case a strategy based on equations (6.6) has been applied. The desired dynamics is provided by a constant or periodic signal associated with the sensorial stimuli. Since 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:

$$\begin{cases} \dot{x} = y - \frac{d_2}{b} f_1(y;k_2;h_2;p_2,q_2) + \sum_i k_{x_i}(x_{r_i} - x) \\ \dot{y} = z + \sum_i k_{y_i}(y_{r_i} - y) \\ \dot{z} = -ax - by - cz + d_1 f_1(x;k_1;h_1;p_1,q_1) \\ \qquad + d_2 f_1(y;k_2;h_2;p_2,q_2) \end{cases} \tag{6.7}$$

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 that will be described in details below 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, and this action is sufficient for the proposed navigation control strategy. The complete control scheme is shown in Fig. 6.2.

Each reference signal (x_{r_i}, y_{r_i}) can be a constant input or a periodic trajectory representing a native cycle. This can be generated using the multiscroll system (6.1) with particular parameters ($a = b = c = 1$). In this case the number of multiscroll systems needed to generate the reference cycles should be the same as the number of reference trajectories required. In a more simple way, these reference signals can be built using sinusoidal oscillators:

$$\begin{aligned} x_r(t) &= A_{x_r} sin(\omega_{xr} t - \varphi_{x_r}) + x_{off} \\ y_r(t) &= A_{y_r} sin(\omega_{yr} t - \varphi_{y_r}) + y_{off} \end{aligned} \tag{6.8}$$

where (x_{off}, y_{off}) is the center of the reference cycle, ω is the 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 signal.

6.2.3 Multiscroll control for robot navigation control

In the following sections the proposed reactive control scheme will be applied to robot navigation control. Taking inspiration from Freeman's works, we adopt a chaos control approach to enslave the chaotic trajectories of our perceptual system towards different pseudo-periodic orbits. We chose to use, as reference signals, periodic inputs, even if the method can be used also considering constant references. To this aim, taking into account a single reference periodic signal, the control gain range is defined in the following way:

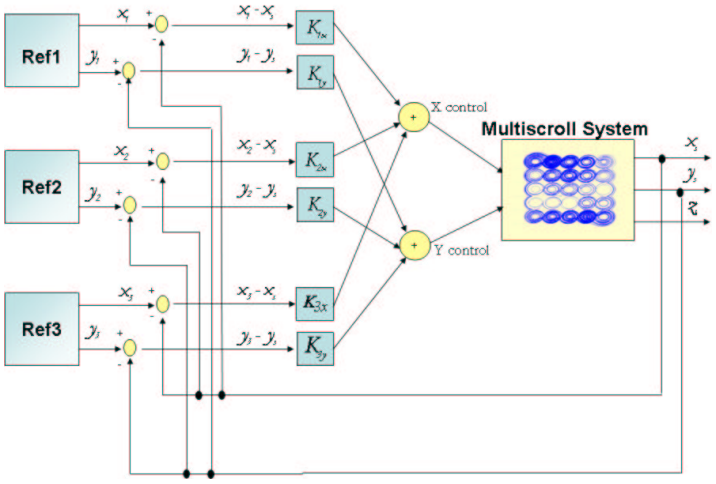
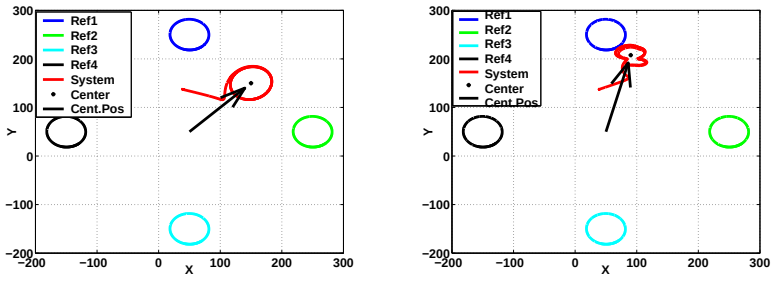


Fig. 6.2 Block diagram of the control scheme when three distinct reference signals (i.e. sensorial stimuli) are perceived by the multiscroll system.

$$k_x, k_y \geq k_{min}(x_r, y_r) \tag{6.9}$$

If the reference signal is a native orbit of the chaotic system, it can be shown that $k_{min} = 0.534$.

Below k_{min} the system presents a chaotic behavior, whereas if control gains are above k_{min} a limit cycle behavior occurs. In particular, as concerns the control by using a single reference dynamics, for low values of k_x and k_y the control of the multiscroll attractor has a residual error, however for the purpose of navigation control this weak condition is still acceptable. For higher values of k_x and k_y, the steady error approaches zero.

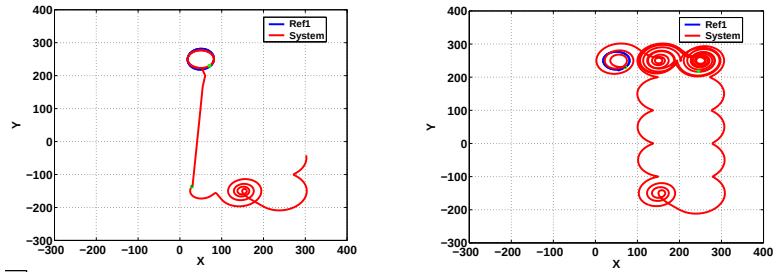


(a) (b)

Fig. 6.3 Limit cycle obtained when the multiscroll system is controlled by two sensorial stimuli. (a) Control gains: $k_{x_1} = k_{x_2} = k_{y_1} = k_{y_2} = 0.8$; (b) Control gains: $k_{x_1} = k_{y_1} = 2$, $k_{x_2} = k_{y_2} = 0.6$.

One of the most interesting aspects of this technique, applied to the multiscroll system into consideration and useful for robot control purposes, is evident when there are more than one external reference. For example, let us consider the case in which there are two concurrently active inputs, and so there are two reference signals in the multiscroll phase plane. If the control gains of the two reference systems are not equal, the resulting controlled limit cycle (*emerged cycle*) will be placed, in the phase plane, near the reference cycle associated with the higher control gain. If the two reference dynamics have the same control gains, the resulting cycle will be placed exactly at halfway between them. These results are shown in Fig. 6.3. When stimuli are perceived, the system converges to a limit cycle that constitutes a representation of the concurrent activation of the sensorial stimuli. When stimuli are no longer active, the multiscroll returns to its default chaotic dynamics. An example of this process is shown in Fig. 6.4.

In the next sections a reactive system applied to the navigation control of a roving robot is proposed.



(a) (b)

Fig. 6.4 An example of the evolution of the multiscroll system when controlled by reference dynamics associated with sensors. (a) When a single stimulus is perceived, the system converges to the reference cycle; (b) when the stimulus ends, the system behaves chaotically.

6.2.4 Robot navigation

To explore an area avoiding obstacles, the robot, sensing the environment, can create an internal representation of the stimuli in relation to its body. The loop is closed by an action that is chosen to accomplish a given target behavior (e.g. exploration with obstacle avoidance). Every sensor equipped on the robot provides a reference cycle. This is addressed by associating for instance, the perception of an obstacle with a stimulus and associating it with a representation (pattern). Therefore the controlled multiscroll system is the perceptual system and the emerged orbit stands for the internal representation of the external environment. Finally, according to the characteristic of the emerged cycle (amplitude, frequency, center position) an action, in terms of speed and rotation angle, is associated. At this stage, action is linked to perception (the emerged cycle) using a deterministic algorithm. However this association can be obtained through a bio-inspired adaptive structure and the classical Motor Map paradigm could represent a good candidate [45]. Fig. 6.5 shows a simulated 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 are associated each one to a single reference cycle, whereas 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 [6]. The offsets assigned to each input cycle are defined to match the position of the scroll centers. Moreover in this work we have chosen to link the value of the control gains with the intensity of perceived sensorial stimuli.

The technique, based on placing reference cycles in the phase plane in accordance with the distribution of sensors on the robot, is important to strictly connect the internal representation of the environment to the robot geometry. In our tests

only distance and target sensors have been used, although other sensors could be included.

Distance sensors have a visibility range that represents the area where the robot is able to detect static and dynamic obstacles, whereas target sensor returns the target angular displacement with respect to the front axis of the robot, when the robot is inside the detection region of the target.

As concerns the action (in terms of absolute value of speed and heading) performed by the robot, it depends on the multiscroll behavior. In particular, when no stimulus is perceived (i.e. there are no active sensors) the system evolves chaotically and the robot continues to explore the environment moving with constant speed and without modifying its orientation. Moreover, another possible exploring strategy can be taken into consideration: the robot can exploit the chaotic wandering of its internal state variables to generate and follow a chaotic trajectory that can help the system during the environment exploration. When external stimuli are perceived, the controlled system converges to a cycle (i.e. a periodic pattern) that depends on the contribution of active sensors through the control gains k_{x_i} and k_{y_i}. The action that will be executed is chosen according to the characteristics of the cycle, in particular its position in the phase plane. A vector pointing to the center of the limit cycle of the controlled multiscroll attractor is defined; predefined actions are chosen on the basis of module and orientation of this vector. When the stimuli stop, the multiscroll returns to evolve in a chaotic way.

A different strategy has been adopted for the target. When a target is in the detection range of the robot, it is considered as an obstacle located in a position symmetric with respect to the motion direction. This is associated with a reference cycle which controls the multiscroll attractor with a low gain, so that avoiding obstacles has priority over reaching targets. In this way the generated reference cycle has the task to weakly suggest a rotation towards the target, since preserving the robot safety is considered more important than retrieving a target.

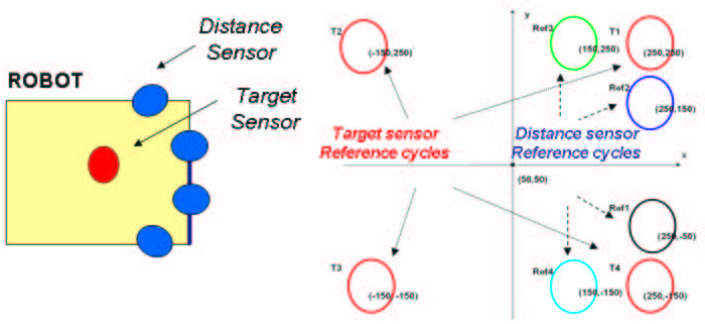


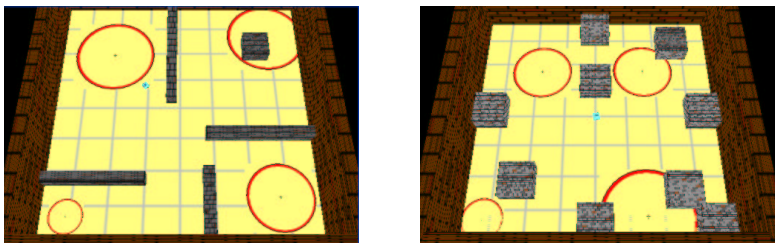
Fig. 6.5 Scheme of a simulated roving robot equipped with four distance sensors and a target sensor. In the phase plane $x-y$, the reference cycles associated to each sensor are reported. Target sensor, due to its omnidirectional field of view, is associated to four reference cycles. Cycles can be generated by system (6.1) with 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, equations (6.8) can be used.

6.2.5 Simulation results

To test the performance and the potential impact of the proposed architecture we developed a software tool for mobile robot simulations and a hardware implementation in an embedded platform. The first evaluation stage was carried out via a 2D/3D simulation environment [6]. A robot model involved in a food retrieval task was simulated.

To evaluate the performance of the proposed control scheme, a comparison with a traditional navigation control method is reported. The navigation strategy chosen as benchmark is the Potential Field (PF) [35]. A basic version using a quadratic potential has been implemented in the simulator for the same simulated robot used to test the weak chaos control approach [9]. Also in this case the robot can use only local information, acquired from its sensory system to react to the environment conditions (i.e. local PF). The parameters of the PF algorithm (e.g. robot speed, constraints for the movements) have been chosen in order to allow a comparison with the WCC technique.

The sensory system of the simulated roving robot consists of four distance sensors and a target sensor as depicted in Fig. 6.5. Several environmental configurations were considered; here we report the results for two different scenarios shown in Fig.6.6. Four targets were introduced in both the environments: the circle around each target represents the range in which the target can be sensed. Obstacles are represented by walls and by the black rectangles. The robot had to navigate in this environment reaching the targets, while avoiding obstacles. When a target is found, it is disabled, so that the robot navigates toward the other targets.



(a) (b)

Fig. 6.6 Environments used to evaluate the performance of the proposed architecture in comparison with the Potential Field. The dimension of both the arenas is 50x50 robot units, the randomly placed objects correspond to walls and obstacles and the circles indicate the visibility range of the targets.

For each environment we performed a set of five simulations for the different control methods taken into consideration. In particular we compared the navigation capabilities of the robot controlled through the local potential field method and

through two versions of the WCC architecture. The difference between the two versions is limited to the behaviour of the robot during the exploration phase (i.e. when no stimuli are perceived). The former implements a very simple behaviour that consists into a forward movement with the speed set to its maximum value (i.e. WCC_f), whereas the latter considers the chaotic evolution of the multiscroll system to determine the action of the robot exploring the environment (i.e. WCC_c). Here when no stimuli are perceived, the perceptual core of the control system behaves chaotically and the robot action depends on the position of the centroid of the chaotic wandering shown by the system during the simulation step. Each simulation step corresponds to a single robot action and it is determined simulating the dynamical system for 2000 steps with an integration step equal to 0.1 [6]. An example of the trajectories followed by the robot in the three cases is shown in Fig. 6.7.

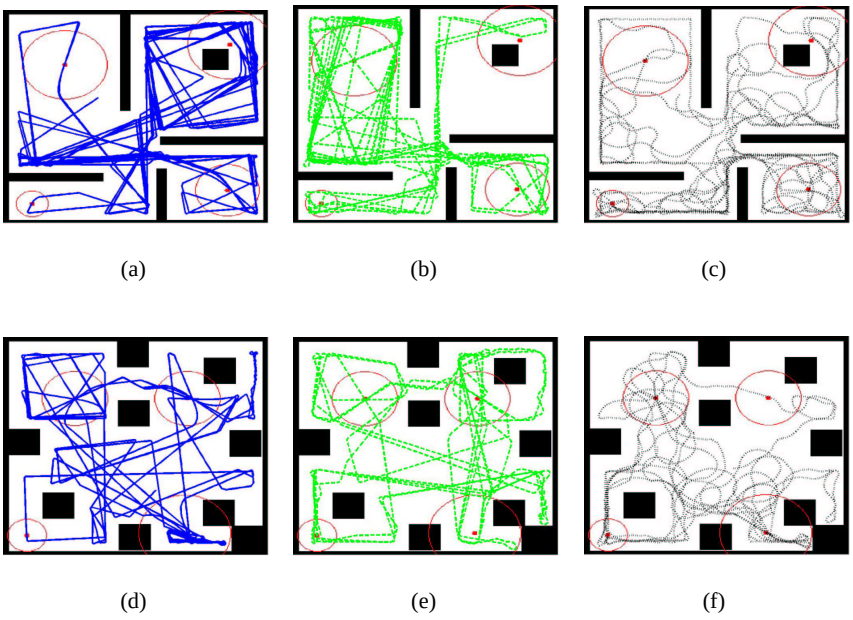


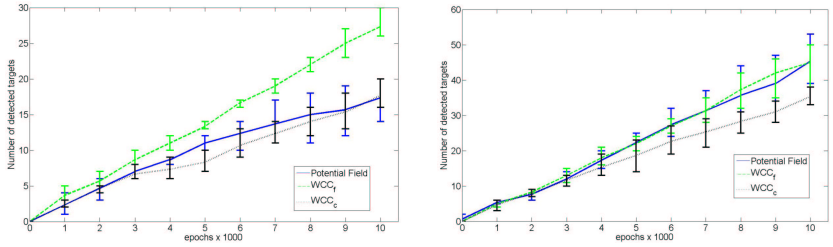
Fig. 6.7 Trajectories followed by the robot controlled through: (a) (d) local potential field; (b)(e) WCC with forward exploration behaviour, (c)(f) WCC with chaotic exploration behaviour.

For each simulation the robot is randomly placed in the environment and the three control methods are applied monitoring the robot behaviour for 10000 actions (i.e. epoches). To compare the performances of the algorithms we consider the cumulative number of targets found and the area explored by the robot [29].

In Fig. 6.8 the cumulative number of targets found in the two environments, calculated in time windows of 1000 epochs, is shown. The performances of the three control methods are comparable in both the environments.

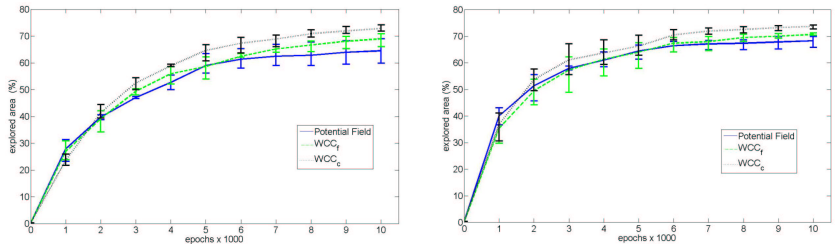
Another performance index taken into consideration is the area covered by the robot during each simulation. The results shown in Fig. 6.9 demonstrate that the WCC_c guarantees a high exploration capabilities with respect to the other control methods.

Movies of these simulations are available on the web [5].



(a) (b)

Fig. 6.8 Cumulative number of targets found for the three control algorithms in the (a) four-rooms environment (Fig.6.6 (a)) and (b) environment filled with obstacles (Fig.6.6(b)). The simulation time is 10000 epochs and the mean number of targets, mediated over 5 simulations, calculated with time windows of 1000 epochs, is indicated. The bars show the minimum and maximum number of targets found.



(a) (b)

Fig. 6.9 Area explored using the three control algorithms in the (a) four-rooms environment (Fig.6.6 (a)) and (b) environment filled with obstacles (Fig.6.6(b)). The arena which dimension is 50x50 robot units, has been divided into locations of 2x2 robot units. The simulation time is 10000 epochs and the mean value of area explored, mediated over 5 simulations, calculated with time windows of 1000 epochs, is indicated. The bars show the minimum and maximum value.

6.3 Learning anticipation in spiking networks

Basic problems in navigation control can be solved using many different approaches [8], our idea is to investigate a suitable navigation control scheme, which relies on learning strategies in spiking neurons [27] (i.e. the fundamental processing units of the nervous system).

Literature on navigation control is vast as well as the papers focusing on biologically inspired approaches to navigation. A review on bio-inspired navigation control schemes is provided by Trullier et al. [50] and by Franz and Mallot [20]. A key role in navigation control in animals is played by hippocampal cells, which encode spatial locations and act as an associative memory [14, 13, 12, 36, 34]. The aim of our work is not to reproduce a biological model, but to use the paradigm of spiking computation to build an artificial structure suitable for the control of a roving robot.

The formulation of the control system in terms of spiking neurons requires to adopt neural mechanisms in order to implement the desired features. For instance, the learning strategy should be based on a biologically plausible mechanism. The so-called Spike-Timing-Dependent-Plasticity (STDP) has been introduced to explain biological plasticity [49, 33]. The same paradigm has been introduced in Chapter 3 but is here summarized for sake of clarity.

According to this biologically plausible theory, synaptic weights among living neurons are changed so that a synaptic connection between two neurons is reinforced if there is a causal correlation between the spiking times of the two neurons. The weight of the synapse is increased if the pre-synaptic spike occurs before the post-synaptic spike, decreased otherwise.

Taking into account the STDP rule, we implemented a system able to learn the response to "high-level" (conditioned) stimuli, starting from a priori known responses to "low-level" (unconditioned) stimuli. To this aim, successive repetitive occurrences of the stimulus lead to reinforce the conditioned response, according to the paradigms of classical and operant conditioning [41, 46], briefly introduced in the following. Classical conditioning relies on the assumption that, given a set of unconditioned responses (UR) triggered by unconditioned stimuli (US), the animal learns to associate a conditioned response (CR) (similar to the UR) to a conditioned stimulus (CS) which can be paired with the US [46]. In operant conditioning the animal is asked to learn a task or solve a problem (such as running a maze) and gets a reward if it succeeds [46]. These two biological mechanisms, used by living creatures to learn from the interaction with the environment, provided the source of inspiration for the implementation of our algorithm.

In our approach learning capabilities are included to implement the main navigation control tasks (obstacle avoidance, target approaching and navigation based on visual cues). These have been implemented through a bottom-up approach: from the most simple task, i.e. obstacle avoidance, to the most complex task, i.e. navigation with visual cues. In the case of obstacle avoidance, contact sensors play the role of unconditioned stimuli (US), while range finder sensors represent conditioned stimuli (CS). In the case of target approach, sensors which are activated in the proximity of a target (called in the following proximity target sensors) play the role of US,

while the visual input coming from a camera equipped on the robot represents the CS. In fact, the robot has *a priori* known reactions for contact and proximity target sensors, but has to learn the response to the conditioned stimuli through classical conditioning. In the case of navigation with visual cues, the development of a layer for visual target approach is fundamental to learn the appropriate response to objects which can become landmarks driving the robot behavior.

Several local navigation tasks can thus be incrementally acquired on the basis of the already learned behaviors. The bio-inspired learning approach is potentially relevant to the goal of progressively including more and more details gained from high-level sensors, like visual cues, into the navigation control loop. New visual features could be used, as learning proceeds, to select more appropriate actions, relevant to the current task.

The network of spiking neurons with STDP learning is applied to control a roving robot. The loop through the world, which defines the input of the network, is exploited to produce classical conditioning in several continuous behavioral scenarios.

6.3.1 The spiking network model

In this Section the mathematical model of the spiking network applied to all the local navigation skills mentioned in the introduction is discussed. This approach derives from the Mushroom Bodies modeling proposed in Chapter 1 and 3 and is here briefly summarized and further extended to include more complex skills. The network consists of interacting spiking neurons, which may play the role of sensory neurons, inter-neurons or motor-neurons. Sensory neurons are neurons connected to sensors, while motor-neurons drive robot motors; inter-neurons are all the other neurons involved in the processing for navigation control. Each neuron is modeled by the following equations proposed by Izhikevich [30]:

$$\begin{aligned} \dot{v} &= 0.04v^2 + 5v + 140 - u + I \\ \dot{u} &= a(bv - u) \end{aligned} \tag{6.10}$$

with the spike-resetting

$$\texttt{if } v \geq 0.03\texttt{, then} \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{6.11}$$

where v, u and I are dimensionless variables representing the neuron membrane potential, the recovery variable and the input current, respectively, while a, b, c and d are system parameters. The time unit is *ms*.

According to the parameters chosen [31], this model could reproduce the main firing patterns and neural dynamical properties of biological neurons, such as spiking behaviors (tonic, phasic and chaotic spiking) and bursting behavior.

Among the possible behaviors class I excitable neurons are selected. In these neurons the spiking rate is proportional to the amplitude of the stimulus [31]. Such property is really important as a way to encode any measured quantity by means of the firing frequency. It also represents a suitable way to fuse sensory data at the network input level. For the task of obstacle avoidance, the robot is equipped with range finder sensors connected to sensory neurons. These are class I excitable neurons able to encode the distance from the obstacle through their firing rate, as schematically shown in Fig. 6.10. For this reason, neuron parameters are chosen as $a = 0.02$, $b = -0.1$, $c = -55$, $d = 6$ (class I excitable neurons [31]), while the input I accounts for both external stimuli (e.g. sensorial stimuli) and synaptic inputs. The same model was adopted for the other neurons of the network.

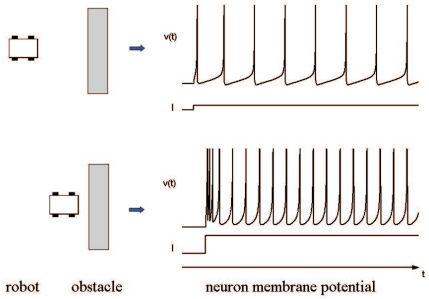


Fig. 6.10 Class I excitable neurons encode the robot distance from the obstacle into their firing frequency.

Concerning the model of the synapse, let us consider a neuron j which has synaptic connections with n neurons, and let us indicate with t_i the instant in which a generic neuron i, connected to neuron j, emits a spike. The synaptic input to neuron j is given by the following equation:

$$I_j(t) = \sum w_{ij} \varepsilon(t - t_i) \tag{6.12}$$

where w_{ij} represents the weight of the synapse from neuron i to neuron j and the function $\varepsilon(t)$ is expressed by the following formula:

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}} & \texttt{if } t \geq 0 \\ 0 & \texttt{if } t < 0 \end{cases} \tag{6.13}$$

Equation (6.13) describes the contribution of a spike, from a presynaptic neuron emitted at $t = 0$ [18]. In our simulations τ has been fixed to $\tau = 5ms$.

To include adaptive capabilities in our model, Hebbian learning was considered. Recent results [49] indicate STDP as a model of experimentally observed biological synaptic plasticity. The synaptic weights of our network are thus allowed to be modifiable according to the STDP rule discussed in [49] and here briefly reported. Let us indicate with w the synaptic weight. A presynaptic spike and a post synaptic spike

modify the synaptic weight w by $w \rightarrow w + \Delta w$, where, according to the STDP rule, Δw depends on the timing of pre-synaptic and post-synaptic spikes. The following rule holds [49]:

$$\Delta w = \begin{cases} A_+ e^{\frac{\Delta t}{\tau_+}} & \texttt{if } \Delta t < 0 \\ A_- e^{\frac{-\Delta t}{\tau_-}} & \texttt{if } \Delta t \geq 0 \end{cases} \tag{6.14}$$

where $\Delta t = t_{pre} - t_{post}$ is the difference between the spiking time of the pre-synaptic neuron (t_{pre}) and that of the post-synaptic one (t_{post}). If $\Delta t < 0$, the post-synaptic spike occurs after the pre-synaptic spike, thus the synapsis should be reinforced. Otherwise if $\Delta t \geq 0$, (the post-synaptic spike occurs before the pre-synaptic spike), the synaptic weight is decreased by the quantity Δw. The choice of the other parameters (A_+, A_-, τ_+ and τ_-) of the learning algorithm will be discussed below. Equation (6.14) is a rather standard assumption to model STDP. The term A_+ (A_-) represents the maximum Δw which is obtained for almost equal pre- and post- spiking times in the case of potentiation (depression).

The use of the synaptic rule described by equation (6.13) may lead to an unrealistic growth of the synaptic weights. For this reason, often upper limits for the weight values are fixed [48, 32]. Furthermore, some authors (see for instance [52, 53]) introduce a decay rate in the weight update rule. This solution avoids that the weights of the network increase steadily with training and allows a continuous learning to be implemented. In the simulations, the decay rate has been fixed to 5% of the weight value and is performed every 3000 simulation steps. Thus, the weight values of plastic synapses are updated according to the following equation:

$$w(t+1) = \begin{cases} 0.95w(t) + \Delta w & \texttt{if } t \texttt{ mod } 3000 = 0 \\ w(t) + \Delta w & \texttt{otherwise} \end{cases} \tag{6.15}$$

where t mod 3000 indicates the modulus of t divided by 3000 and Δw is given by eq. (6.14).

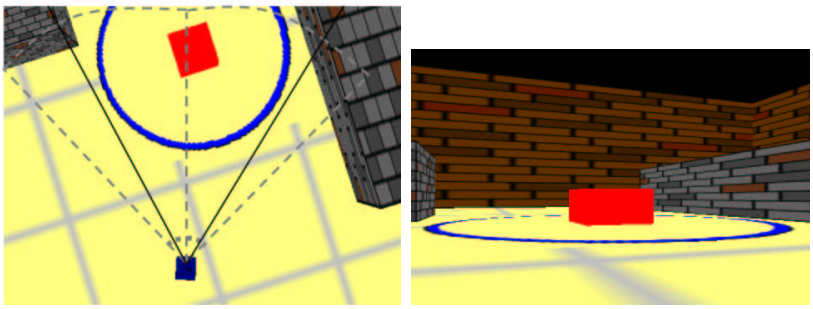
In the following, the upper limit of the synaptic weight was fixed to 8 (excitatory synapses) or -8 (inhibitory synapses). The initial values of synapses with STDP are either 0.05 (excitatory synapses) or -0.05 (inhibitory synapses). All the other synaptic weights (i.e., weights not subject to learning) are fixed to their maximum value.

6.3.2 Robot simulation and controller structure

In this Section the simulated robot, the simulation environment and the interface between the spiking network, the robot and the environment are briefly described.

The simulated robot has a cubic shape: in the following the edge of the robot will be used as measure unit, and indicated as robot unit (r.u.). The simulated robot is a dual-drive wheeled robot: the two motors are labeled as left (*LM*) and right motor (*RM*). The two actuated wheels are driven by the output of the spiking net-

work structure as discussed below. The robot is equipped with two collision sensors, two range finder sensors, two proximity sensors for target detection and a simulated visual camera. Contact and range finder sensors are centered on the front side of the robot and have a range of $[0^\circ, +45^\circ]$ (right sensor) or $[0^\circ, -45^\circ]$ (left sensor). The proximity sensors for target detection are range finder sensors which sense the presence of a target. The simulated robot is shown in Fig. 6.11: a schematic representation of sensor ranges is given in Fig. 11(a), while in Fig. 11(b) the on-board camera view is shown.



(a) (b)

Fig. 6.11 (a) The simulated robot is equipped with range finder sensors centered on the front side of the robot and with a range of $[0^\circ, +45^\circ]$ (right sensor) or $[-45^\circ, 0^\circ]$ (left sensor), schematically shown with gray dashed lines (outer arcs). The inner arcs refer to the range of contact sensors. The range of the on-board camera view is shown with black solid lines. The circle surrounding the target (the red object) represents the range in which the target is sensed by the proximity target sensors of the robot . (b) On-board camera view.

The general structure of the spiking network is made of three layers of neurons outlined below.

The first layer is formed by sensory neurons. We used class I excitable neurons to model this layer of neurons, so that the frequency rate codes the stimulus intensity. The second layer is formed by inter-neurons, whereas the third layer is formed by motor-neurons. The output of the motor-neurons is used to generate the signals driving the two wheels of the robot. We now briefly clarify how each layer works in general.

The response of the sensory neurons depends on the stimulus intensity through the input I in equation (6.10). Let us firstly consider sensory neurons associated with collision sensors and let us indicate with d_0 the distance between the robot and the closest obstacle. This distance is computed in robot units (r.u.). Collision sensors are activated when the distance d_0 is $d_0 \leq 0.6 r.u.$: in this case the input of the neuron associated with the US is fixed to a constant value $I = 9$. This value is such that the

sensory neuron emits regular spikes which drive the robot to avoid the obstacle by turning in one direction.

In the case of range finder sensors for obstacle detection, the input is a function of d_0: $I = 9e^{-0.6d_0} + 2.2$. This function has been chosen so that the range finder sensors approximatively begin to fire when the distance between obstacle and robot is less than 11 r.u. In the case of range finder sensors for target detection, the same input function is used, but the target can be detected if the distance from the robot is less than 5 r.u.

As far as the visual sensor is concerned, input is acquired by a camera equipped on the simulated robot. The raw image (388x252 pixels) is pre-processed in order to identify objects of different colors. In particular, to develop the network for target approaching and navigation with visual cues, two color filters have been considered: red objects represent targets whereas yellow objects are used as visual cues. In both the cases the filtered image is tiled in 9 different sectors, each one associated with a class I spiking neuron. A visual sensory neuron is activated if the barycenter of the identified object falls within its corresponding sector. The input is a function of the perimeter of the object, as follows. Let us indicate with p the perimeter of the biggest object in the visual field, then the input I of the active sensory neuron is given by: $I = 0.012p + 3.8$ if $0 < p \leq 400 pixels$ or $I = 8.6$ if $p > 400 pixels$.

Inter-neurons constitute an intermediate layer between sensory neurons and motor-neurons. We found that this layer is not strictly necessary in the avoidance task where the overall network is quite simple, but is needed for more complex tasks.

Finally, the last layer of the network is formed by four motor-neurons. For uniformity these neurons (and the inter-neurons as well) are also class I excitable neurons. Motor control is strictly connected to the robot structure under examination. To clarify how the output of the motor-neurons is used to drive the robot wheels, we need to briefly mention some aspect of the simulation.

The robot moves in a simulated environment filled with randomly placed obstacles. At each simulation step, a time window of $300ms$ of model behavior is simulated. The robot moves according to the number of spikes generated by the motor-neurons during this time window. The number of spikes emitted by the two left (right) motor neurons are cumulated to compute the robot action, after that a new sensory acquisition and network processing will be executed.

In our model we assume that the speed of the motor is proportional to the spiking rate of the driver signal. The driver signal for each motor depends on the number of spikes in the variables v (eq.6.10) of the associated motor-neurons. Referring to the spiking network shown in the box of Fig. 6.12, the “go-on” motor-neurons generate the spike train needed to let the robot advance in the forward direction (even in absence of stimuli). The variables v of the other motor-neurons (since this part of the network is needed to make the robot able to turn, we refer to these neurons as “turn” neurons) are then summed to the ones of the “go-on” neurons, so that the driving signals of the motors are the sum of the two spike trains. In presence of collisions, the network structure is such that the “go-on” motor-neurons are inhibited and the forward movement is suppressed. When the left and right motor-neurons emit an

equal number of spikes, the robot moves forward with a speed proportional to the number of spikes. In absence of conditioned stimuli the amplitude of the forward movement is about 0.3 r.u. for each step.

When the information gathered by the sensory system produces a difference in the number of spikes emitted by left and right motor-neurons, the robot rotates. Let n_R (n_L) be the number of spikes in the signal driving the right (left) motor, and let $\Delta n_s = n_R - n_L$, then the angle of rotation (in the counterclockwise direction) is $\theta = 0.14\Delta n_s$ rad. This means that if for instance $n_R = 5$ and $n_L = 4$, then the difference between the number of spikes emitted by right and left motor-neurons is one spike and the robot will rotate of 0.14 rad (about 8°). In this case, after the rotation the robot proceeds forward. We count the spikes emitted both by the left and the right neuron (i.e., the minimum value of n_R and n_L; in the example 4) and let the robot advance with a speed proportional to this number, so that the spike rate codes the robot speed.

Without learning, the robot is driven by unconditioned behavior. In this case, the robot is driven by contact sensors and is able to avoid obstacles only after colliding with them. Furthermore, thanks to proximity target sensors the robot can approach a target only if this is within the sensor range. In the case of a front obstacle, the turning direction is random.

6.3.3 Spiking network for obstacle avoidance

The spiking network described in this Section deals with the navigation task of moving through an environment where obstacles are randomly placed. Navigation takes place by using fixed reactions to contact sensors (i.e. to US) and learning the appropriate actions in response to conditioned stimuli (i.e. stimuli from range finder sensors).

The network for obstacle avoidance is shown in Fig. 6.12. It is functionally divided in two parts. The first group of neurons (included in the grey box) deals with unconditioned stimuli US_R and US_L (i.e. contact sensors). Synaptic weights of this network are fixed and represent the *a priori* known response to unconditioned stimuli, i.e. the basic avoidance behavior of the robot. The structure of the connections between neurons is modeled with direct inhibition and cross-excitation among sensory and motor neurons. This model is biologically relevant: in fact it has been used as model for cricket phonotaxis in [55]. Details on cricket phonotaxis and spiking networks are reported in Chapter 3. Concerning sensory neurons, we assume that the obstacles are detected by contact sensors at a distance that allows the robot to avoid the obstacle by turning in one direction.

The second part of the network, whose connections are outlined in gray in Fig. 6.12, deals with conditioned stimuli. The synaptic weights of this network evolve according to the STDP rule discussed above. The synaptic weights of neurons connected to CS_L and CS_R are not known a priori and should be learned dur-

ing the exploration. The following parameters have been used for the STDP rule: $A_+ = 0.02$, $A_- = -0.02$, $\tau_+ = 20ms$, $\tau_- = 10ms$.

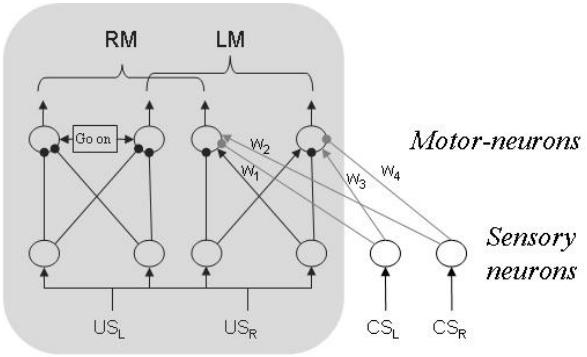
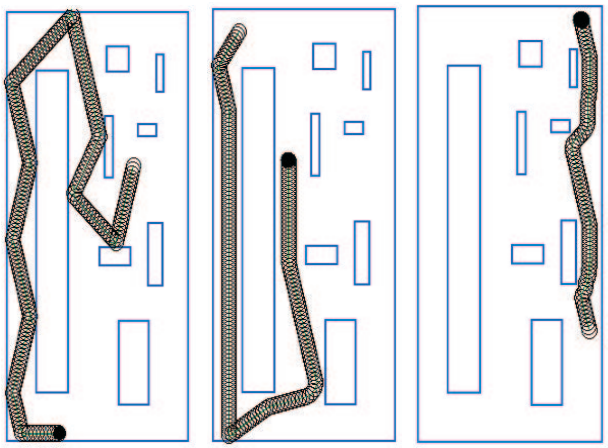


Fig. 6.12 Network for obstacle avoidance. *RM* and *LM* are right and left motors, respectively. US_R and US_L represent right and left contact sensors, respectively, while CS_R and CS_L represent conditioned stimuli (i.e. range finder sensors). Each circle indicates a class I excitable neuron modeled by equations (6.10). Arrows and dots indicate excitatory and inhibitory synapses, respectively. The 'go on' box indicates a constant input which makes the motor neurons fire in the absence of sensory input (this constant input is such that the robot can go in the forward direction without sensory input).

Without stimuli, the motor-neurons controlling right and left motors emit the same number of spikes and the robot proceeds in the forward direction. When a stimulus, due either to a contact or a distance sensor, occurs, the number of spikes emitted by the motor-neurons are no longer equal and a steering movement occurs.

To illustrate a first example of obstacle avoidance, we take into account two parameters. A steering movement is the response of the robot to an external stimulus. It can be either an unconditioned response (UR) or a conditioned response (CR). To test the behavior of the network controller, we take into account how many steering movements are due to an US or to a CS. We define N_{US} as the number of avoidance movements (i.e. steering movements directed to avoid an obstacle) which occur in a given time window and are due to the triggering of an UR. We define N_{CS} as the number of avoidance movements in a given time window, due to the triggering of a CR. If the spiking network controller performs well, N_{CS} grows, while N_{US} decreases.

Trajectories generated in a typical experiment are shown in Fig. 6.13. During the first phase the robot avoids obstacles by using UR, triggered by contact sensors (Fig. 13(a)). During this phase the robot learns the correct CR. Proceeding further with the simulation (Fig. 13(b)), obstacles are finally avoided only by using range finder sensors, i.e. CS (Fig. 13(c)). The weights converge towards an equilibrium value given by the balance between new experience, learned when collision sensors are activated, and the weight decay rate.



(a) (b) (c)

Fig. 6.13 Three meaningful parts of a simulation devoted to learn CRs. (a) Robot behavior is driven by UR. (b) The robot has already learned CR to left obstacles. After two collisions with right obstacles, it avoids a right obstacle with a CR. (c) The robot avoids obstacles using CR. The final position, for each of the simulation phases, is indicated with a filled circle.

The introduction of the weight decay rate implies a never ending learning. This is not a drawback, since the learning algorithm is simple, it can run in real-time, and a continuous learning contributes to improve experience and also to face with dynamically changing environments.

In Fig. 6.14 the number of avoidance movements due to CS (namely N_{CS}) is compared with the number of avoidance movements due to US (namely N_{US}). The time window in which N_{US} and N_{CS} are calculated is 1000 simulation steps. As it can be noticed, as new experience is gained, US are seldom used, and the robot uses CR to avoid obstacles.

We observed several emergent properties in this simple network controller. First of all, it is worth noticing that, although the UR network and the structure for the CR network are symmetrical, the weight values of the CR network are not symmetrical. This provides to the robot a decision strategy in case of front obstacles. In this sense the behavior of the network depends on the configuration of obstacles used to train the network and the robot starting point. The asymmetry depends on how many left or right obstacles are encountered during the first phase of the learning. Other simulation tests, dealing with new obstacles placed in the environment after the learning phase, reveal that the new obstacles in the trajectory of the robot are avoided without any problem. Thus, although the learned weight values depend on the obstacle configuration, the global behavior of the network is independent of obstacle positions.

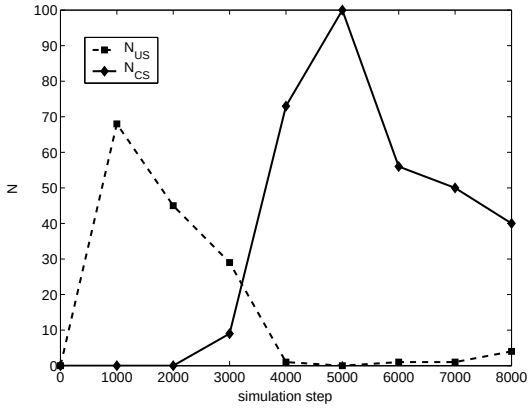


Fig. 6.14 N_{CS} compared with N_{US} in the case of obstacle avoidance. The time window in which N_{US} and N_{CS} are calculated is 1000 simulation steps.

Thanks to the introduction of a decay rate in the synaptic weights, the learning phase never stops. This provides the robot with the capability of redefining, during the time, its turning strategies. Let us focus on the example shown in Fig. 6.15. When the robot approaches the arena wall at the area marked with B, it turns to the right, while an optimal strategy would require to turn in the opposite direction. In B the robot performed a wrong turn, but in D the turn is correct. So, during the time needed to go from B to D, the robot weights were adjusted. This occurs during the four right turns between B and D and, in particular, during the left turn marked as C where collision sensors (i.e. evoking an UR) are used. In fact, with the strategy learned in B the robot collides at point C (because the solution is not optimal) and so it learns a new solution. The robot applies the new solution at point D.

The emergent result is that the robot is able to correct the not optimal strategy and to adopt a better strategy. In fact, when it approaches area marked with D it now turns in the left direction. These emergent properties derive from the dynamic behavior due to the presence of learning (i.e. to the plasticity of synapses of dynamical spiking neurons).

To evaluate in details the performance of the spiking network controller, a systematic analysis was carried on. In particular, an extensive set of simulations were carried out, whose detailed report can be found in [7].

6.3.4 Spiking network for target approaching

6.3.4.1 Spiking model

The first basic mechanism needed for navigation control is the ability to direct towards a target. We suppose that the robot knows how to direct itself towards the

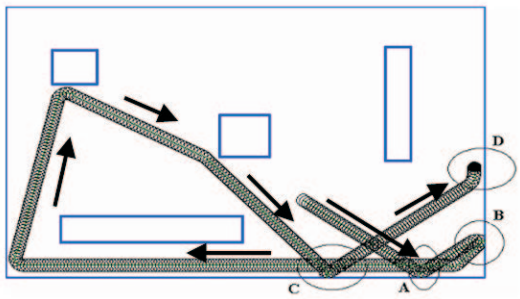


Fig. 6.15 The robot controlled by the spiking network for obstacle avoidance is able to reorganize its turning strategy. The final point of the trajectory followed by the robot is indicated with a filled circle.

target once the target is within a circle of a given radius. However, this a priori known mechanism provides only a poor target approaching capability useful for very low distances: the robot should learn how to direct to the target on the basis of the visual input, which in this case constitutes the conditioned stimulus. In our case, targets are small red cubic objects.

In order to focus on the problem of learning the conditioned response to the visual stimulus, we simplify the network of neurons devoted to process the visual input. We assume that the visual input provides high-level information: red objects are identified and information on their perimeter and their barycenter is provided to the network. The network of neurons is a topographic map of 9 neurons which activate when the object barycenter is within the spatial region associated with them. The input image is thus divided in a regular grid of 9 compartments which define the receptive field of each neuron.

The network of spiking neurons, including visual target approaching, is shown in Fig. 6.16. Unconditioned stimuli for target approaching are indicated as T_L and T_R (left and right): they are range finder sensors, which detect the distance from the target if the robot is within the proximity range outlined in Fig.11(a). The whole network is now divided in three layers: sensory neurons, inter-neurons and motor-neurons. In addition to the plastic synapses of the obstacle avoidance layer, the weights of the synapses between sensory visual neurons and inter-neurons are also modulated by the STDP rule. The other weights are fixed.

It has been assumed that the visual input is the result of a segmentation algorithm through the graphical interface of the simulator, which is solved at the level of sensory neurons. This can be assumed to be feasible, since, with current technology, it is possible to draw image segmentation within the interframe rate [2].

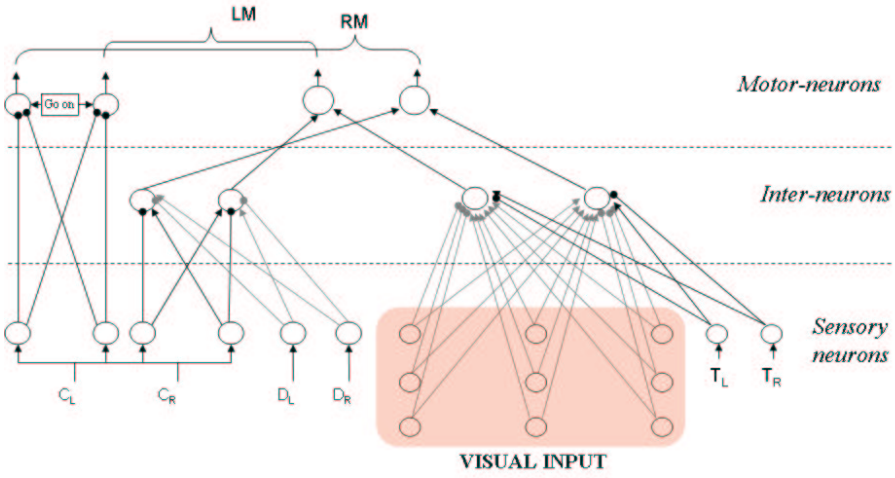


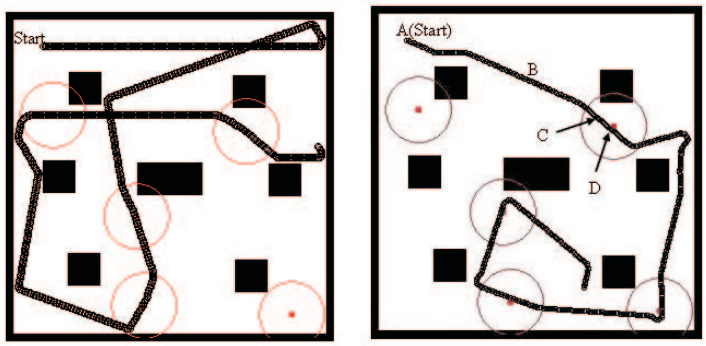
Fig. 6.16 Network for target approaching based on visual input. D_L, D_R, C_L and C_R indicate distance/contact left/right sensors. They have been renominated with respect to Fig. 6.12, in which they were indicated as CS_L, CS_R, US_L and US_R, because now unconditioned stimuli are represented by proximity target sensors (T_L and T_R) and the conditioned stimulus is the visual input.

6.3.4.2 Target approaching: simulation results

In this Section, simulation results focusing on target approaching are shown. We describe several simulations carried out to test the behavior of the robot in a generic environment with randomly placed obstacles and in specific cases, when for instance there is a target difficult to find. In any case, when there are multiple targets in the environment, to avoid that the robot stops at a target, once it is reached, this target is deactivated. The target is then reactivated when a new target is reached.

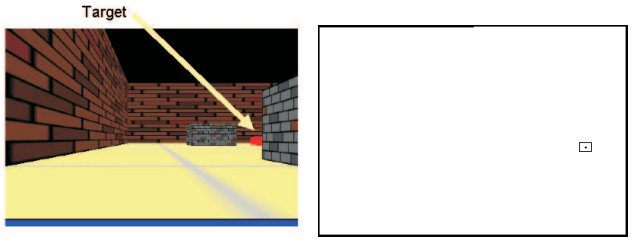
The first experiment refers to the environment shown in Fig. 6.17. The trajectory followed by the robot in this environment in the first phase of the simulation (i.e. when the robot has not yet learned to use visual input) is shown in Fig. 17(a). After learning, the robot follows the trajectory shown in Fig. 17(b). In the case of Fig. 17(b) instead of proceeding in the forward direction as in Fig. 17(a), at point A the robot rotates since it sees the target that it will reach at point D. This is shown in Fig. 6.18, which reports the robot camera view at point A and the visual input: the target is visible on the right, hence the robot follows a different trajectory with respect to the case of Fig. 17(a). Figure 6.19 compares the number of targets found when the network is trained and when it is not trained. Clearly, visual target approaching allows a greater number of targets to be found.

A long run simulation is shown in Fig. 6.20, where there are six targets and five obstacles. The whole trajectory of the robot is shown. It can be noticed that, after learning the target approaching tasks, the robot follows a quite stereotyped trajectory which allows it to visit all the targets avoiding the obstacles. It is worth noticing that,



(a) (b)

Fig. 6.17 Trajectory followed by the robot with visual target approaching control: (a) without learning; (b) after learning. Targets are indicated with small red rectangles inside a circle representing their visibility range.



(a) (b)

Fig. 6.18 (a) Robot camera view at point A of Fig. 17(b). (b) Visual input: the small rectangle indicates that the red object has been identified as a target.

in this case, on the long run, the robot has learnt an optimal trajectory to visit all the targets.

To further test the behavior of the network controller for target approaching, a simulation environment (shown in Fig. 21(a)) with only two targets is considered. Since a target is deactivated once reached, at each time step only a target is active and the robot is forced look for it. Since one of the two targets is hidden by three walls, this simulation allows us to investigate the behavior of the robot when there is a target difficult to be found without visual targeting. The comparison between targets found with and without learning, shown in Fig. 21(b), demonstrates the efficiency of learned visual feedback. It can be observed that after learning the robot is able

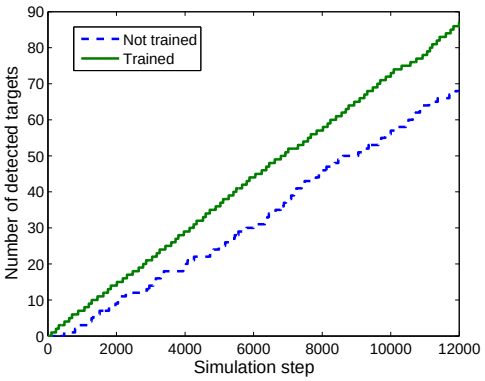


Fig. 6.19 Number of targets found with or without visual input (environment of Fig. 6.17).

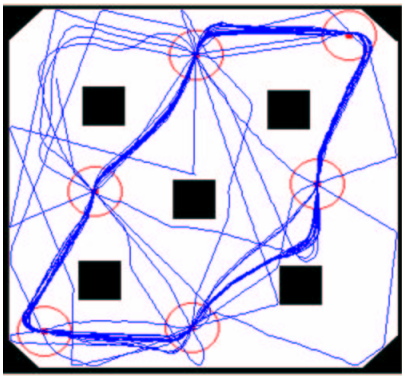
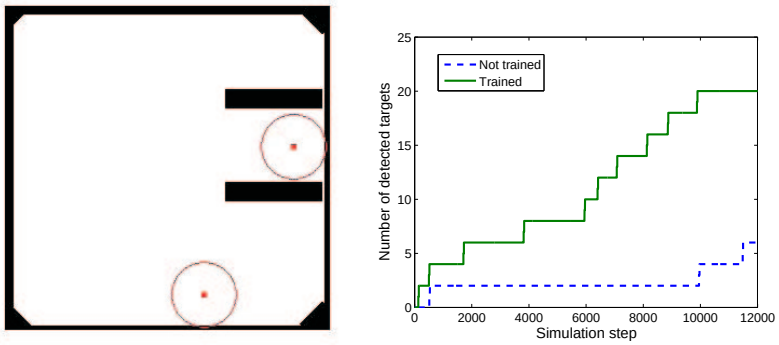


Fig. 6.20 Long run simulation of target approaching showing how at the end of the simulation the robot follows a stereotyped trajectory which allows it to visit all the targets avoiding the obstacles.

to find the targets more easily. In fact, the robot, with range finder sensors, trained without visual input, has to wander for long time before approaching the target: as it can be noticed in Fig. 21(b) before learning (dashed curve) the robot for a long time interval ($500 < t < 10000$ simulation steps) does not find the target hidden by the two walls.

6.3.5 Navigation with visual cues

In this Section the use of a network of spiking neurons for navigation based on visual cues is investigated. The objective is to learn the appropriate response to a given visual object which then can become a landmark driving the robot behavior. We refer to the animal behavior observed in experiments in which rats have to find a food reward in a T-maze. In such experiments [50], depending upon training conditions and on "what is invariant throughout the trials" [44], different cognitive responses can be observed: if the food is always in the right arm of the maze, the animal learns



(a) (b)

Fig. 6.21 (a) Environment referred to the second set of simulations for target approaching. (b) Number of targets found with or without visual input.

an egocentric motor response (i.e. a right-turn response which does not depend on the environmental cues); if the two arms of the maze are physically distinguishable and the cue position is always correlated with the food position, the animal learns to rely on the cue to choose the turning direction; finally, the animal can also learn to rely on extra-maze cues with a response which is triggered by place recognition.

In particular, we take into account a T-maze in which the target (again a red object) is in one of the arms of the T-maze, and a yellow object is positioned on the front wall in slightly different positions. Depending on the position of this yellow object and of the target, the yellow object can be considered a meaningful or useless visual cue.

The scheme of the spiking network for this navigation task is shown in Fig. 6.22. There are now two layers of class I excitable neurons which process the visual input. In particular, the first layer (bottom, left-hand side in Fig. 6.22) exactly acts as in the network of Fig. 6.16: neuron spikes code the presence of a red object in the receptive field of the neuron. We assume that it is a priori known that red obstacles are targets, i.e. we assume that the visual targeting association task has been already solved. The second layer of neurons acts in parallel with the first one. Neuron spikes now code the presence of a yellow object in the receptive field of the neuron. The neurons belonging to this layer are the focus of training and the synaptic connections between the neurons of this layer and the inter-neurons are updated according to the STDP synaptic rule. By updating these synaptic weights the robot has now to learn the appropriate turning direction in correspondence of a given yellow object. A synaptic weight is increased when after a turn the robot sees the target.

We focus on different training environments, based on T-mazes, in order to learn: visual cue-based response or egocentric motor response.

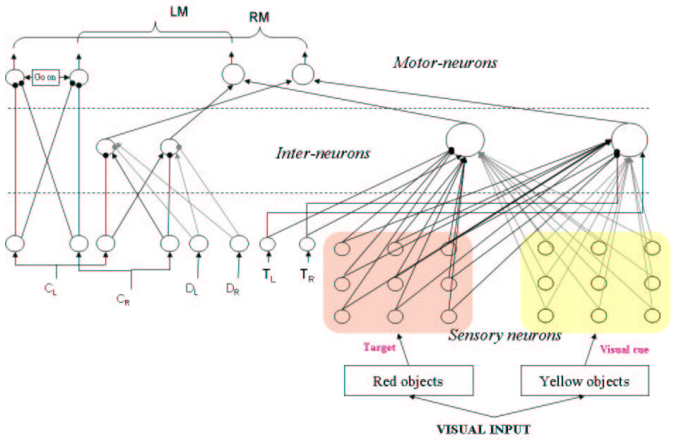


Fig. 6.22 Scheme of the spiking network for navigation with visual cue.

In correspondence of the T bifurcation there are yellow marks placed in the front wall. These objects are landmark candidates, since they can became landmarks if, after successive presentations of the environment configuration, the robot discovers a relationship between the yellow mark and the target position.

As occurs in animals, a given cue/target relationship can be learned if the animal (robot) faces the situation several times. For this reason, the learning protocol consists of several training cycles, Each trial stops either when the robot reaches the target or the end of the incorrect arm. The first case occurs when the robot has performed the right choice at the T bifurcation, while the opposite holds for the second case, in which the robot does not find the target. At the beginning of the training the robot finds the target in the 50% of the trials. During the first 10 training cycles, the robot's ability to find targets does not change significantly. After 13 training cycles, the robot has already learned the association and finds the target in all the following trials. However, in this case the robot turns in proximity of the T bifurcation. If learning proceeds (in particular after 16 training cycles), the robot learns to turn before reaching the T bifurcation (as shown in Fig. 6.23).

The navigation abilities of the robot after learning were further evaluated on a maze in which the target can be found by following visual cues. These cues are placed in the opposite side with respect to the correct arm. We let a simulated robot to navigate in this maze; the simulated robot has been learnt in the simulation environment of Fig. 6.23 which takes into account the same rule needed to solve the maze. As shown in Fig. 6.24, the robot is able to successfully navigate in the maze and find the target.

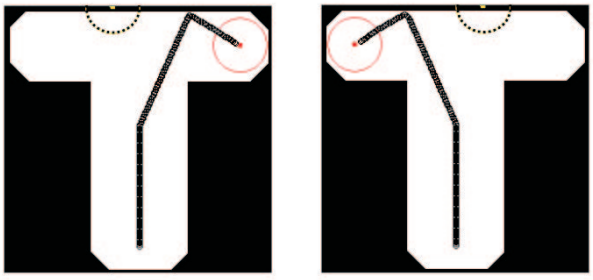


Fig. 6.23 Trajectories followed by the robot for two different cases. The robot learns to turn in the opposite side with respect to the landmark.

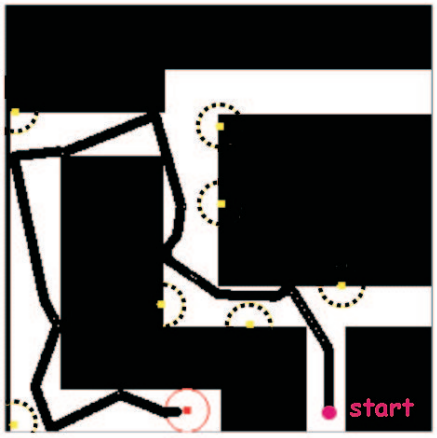


Fig. 6.24 Trajectory followed by the robot in a maze which can be solved relying on visual cues. Visual cues are oppositely placed with respect to the right choice as in the training environment of Fig. 6.23.

6.4 Application to landmark navigation

As already discussed, either for animals or for artificial agents the ability to navigate is crucial in order to move autonomously in the surrounding environment. Many animals have proved to be very good at navigating through significant places using a combination of different strategies in their environment, despite of their simple brain structure [15, 40, 16].

In this section, we applied the reactive model for navigation and the STDP learning rule to create correlation among stimuli in order to formalize an *homing* strategy: homing is a term that robotics has borrowed from biology [19, 56]. It is usually used to describe the ability of various living beings to come back to their nest once accomplished other tasks, like foraging.

Experimental results have shown that animals, in particular insects and rodents, apply geocentric coordinate systems in navigating [57, 10, 1]: ants and bees, for example, use a sun compass system exploiting the polarization pattern of the sky. In this way, they build a coordinate system adopting an absolute direction. For land-

mark navigation rodents seem to use knowledge of distance and direction relative to the landmarks, whereas insects are supposed to receive this information in an indirect way, namely, by image matching. From a biological point of view, the study of insect navigation is subdivided according to the different strategies and modalities used by insects for the homing task. Two of the primary modalities are path integration (i.e. dead reckoning) and the use of visual landmarks. Path integration is the application of orientation and odometry to determine the distance and the phase relative to the target position, e.g. the nest. To correct the loss of precision due to this *open loop* mechanism, especially in proximity of the nest, insects rely on the visual field, matching the current visual pattern with that one which they have previously stored in memory [58]. In general, two paradigms have emerged to describe visual homing in insects. Möller defines these as the template and parameter hypotheses. The template hypothesis [15] assumes that an image taken from the goal is stored as the representation for that position. This image is fixed at the retinal positions at which it was originally stored. The parameter hypothesis [1] assumes that some set of parameters is extracted and stored as the representation for the goal position. According to the terminology of Franz and Mallot, this type of navigation is called *guidance*, which consists in finding a nonvisually marked location using knowledge concerning its spatial relation to visible cues.

The problem of determining which of the visual cues present in the environment could be considered as reliable landmarks has been addressed by processing the external information through a simple network of spiking neurons with dynamic synapses plastically tuned by an STDP algorithm: it allows a synaptic connection between two neurons to be reinforced if there is a causal correlation between the spiking times of the two neurons. In particular, the learning process establishes correlations between the incoming stimuli representing features extracted from the scenario and the nest. This kind of approach has already been used to model the paradigm of classical conditioning with a system able to associate the correct response to high-level (conditioned) stimuli, starting from a priori known responses to low-level (unconditioned) stimuli [7]. To this aim, successive repetitive occurrences of the stimulus lead to reinforce the conditioned response, according to the paradigms of classical and operant conditioning [41, 30]. Once established the landmarks, the agent acquires the geometric relationships which hold between them and the goal position. This process defines the parameters for a recurrent neural network (RNN) which drives the robot navigation, filtering the information about landmarks given within an absolute reference system (e.g the North) in presence of noise. The task of the network investigated in this work is to find a location S starting from any position within the (two-dimensional) workspace. This location is not visually marked, but its position relative to other, visible landmarks is known. When the absolute reference is not available, a safety mechanism acts to control the motion maintaining a correct heading.

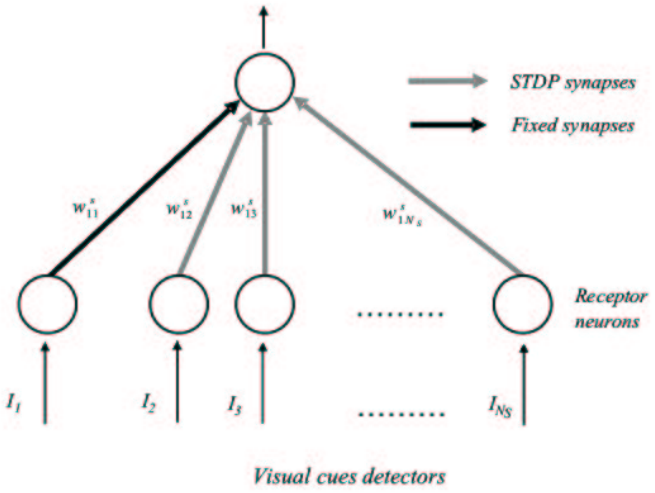


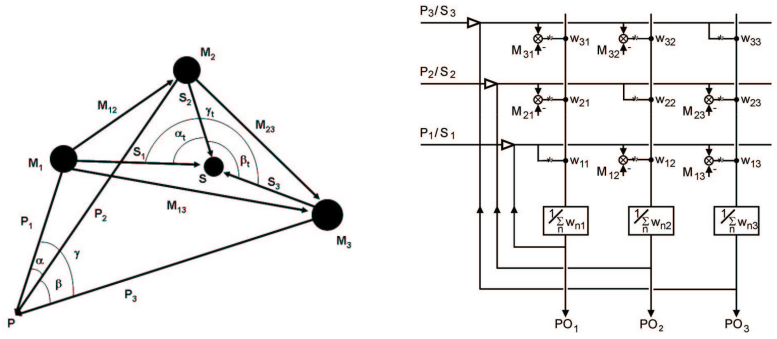
Fig. 6.25 Network for landmark identification in the case of $N_i = 1$: the arrows at the bottom of this figure represent the stimuli (I_i) from the sensor detecting the presence of the target ($i = 1$) and of the visual cues ($i = 2,\ldots,N_S$). Each circle indicates a class I excitable neuron. The connection in black between neurons indicates that the corresponding weight is fixed while the connection in gray indicates that the corresponding weights are plastic, according to the STDP learning rule.

6.4.1 The spiking network for landmark identification

The first phase of the proposed algorithm is devoted to landmark identification. The mathematical model of the network of spiking neurons responsible for the landmark identification. For this purpose, at each time step of the agent, we assume to have information about the presence of some specific visual cues within the navigation environment, thanks to some pre-processing mechanism of the visual stimuli. Each different visual cue, extracted during the time evolution, is considered as a landmark candidate and, if repetitive presentations in proximity of the target occur, the agent recognizes that it is stable in time and space and thus it can be considered as a landmark.

The network model consists of two layers of spiking neurons. The first layer is made of N_S neurons: the first neuron is activated by the receptor for the target detection. The second group of the first layer plays the role of sensory receptor for the different visual cues: in particular, each neuron is activated by the detection of the presence of a specific visual cue. The second layer is made of N_i inter-neurons (in our model $N_i = 1$) used for setting up the correlation between visual cues and the target, i.e. the nest (Fig. 6.25). Each neuron is modeled by the Izhikevich neuron model [30]:

Class I excitable neurons are selected, since the frequency rate of class I excitable neurons may be used to encode any measured quantity by means of the firing frequency, in order to fuse sensory data at the network input level (neuron parameters are chosen as $a = 0.02$, $b = -0.1$, $c = -55$, $d = 6$).



(a) (b)

Fig. 6.26 (a) Spatial map of three landmarks M_1, M_2, and M_3, a source location S, and the actual position of the agent P. Vectors M_{nm} connect the landmarks, vectors S_n connect the source with the landmarks, and vectors P_n connect the actual position of the agent with the landmarks. (b) The recurrent neural network for navigation. Only the net for one component (x or y) is shown. Input values are $P_n(k)$ or S_n; output values are $P_n(k+1)$. The harmony values H, used to determine the network performance, are determined by a separate system (dashed lines). Stars symbolize summation of squared values. See text for further explanation.(The figure is reported from Chapter 4).

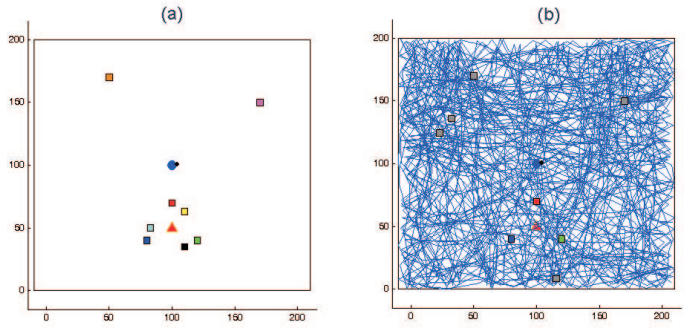


Fig. 6.27 a. Schematic configuration of navigation environment. The triangle symbolizes the target, stable visual cues are represented with red, green, blue, orange and purple rectangular objects, while objects in black, yellow and cyan represent visual cues randomly placed at each iteration. b. Route of the agent during the environment exploration phase. The visual cues not recognized as stable landmarks are discarded in the successive map creation phase.

In our network, we fixed the weights related to the neuron activated by the target: in particular $w^s_{11} = 8$. The other synaptic weights w^s_{1j} ($j = 2,\ldots,N_S$) are allowed to be plastic according to the STDP rule previously introduced. In our simulations, parameters A_- and A_+ are fixed at values $A_- = -0.02$, $A_+ = 0.02$, while $\tau_+ = 20$ and $\tau_- = 10$.

6.4.2 The recurrent neural network for landmark navigation

In order to solve the guidance task, we implemented the recurrent neural network proposed by Prof. H. Cruse and introduced in Chapter 4. In order to describe how this network can be applied to a navigation task, as illustrated in a previous work [17], we assume that the agent knows the relative position of landmarks M_n ($n = 1...N_l$, in the example we use $N_l = 3$) and of an additional, not visually marked, location S, the location of a food source (see Fig. 26(a)).
This knowledge has been acquired through earlier learning and is stored in the form of vectors M_{nm}, pointing from landmark n to landmark m, and the vectors S_n, pointing from landmark n to the source S. Furthermore, we assume that the agent is able to determine N_l vectors $P_n(k)$ that point from each landmark M_n to the actual position P of the agent. The complete network consists of two individual, independent sub-networks, one for the x cartesian components and the other for the y cartesian components of vectors (Fig. 26(b)) built using the North direction as the y-axis and one of the landmark as the origin. Each sub-network contains three units or neu-

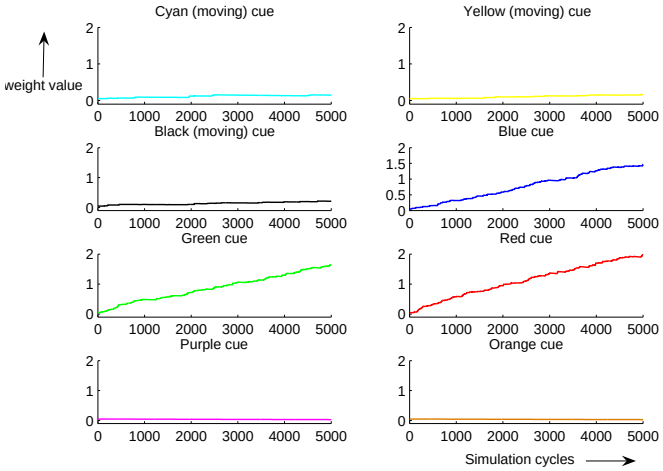


Fig. 6.28 Evolution of synaptic weights during simulation steps. Visual cues that are stable in time and in space and near the target have weights values over the fixed threshold = 1.5 while the other visual cues have no significant synaptic values.

rons $P_n(k)$ ($n = 1,2,3$). The scalar values M_{nm} represent the components stored in memory, while the initial condition $P_n(0)$ are set by the current measurement of the agent-landmarks distances. The following equations show the mathematical structure of the model; they represent a linear model in which weights w_{nm} are arbitrary but fixed.

$$P_1(k+1) = \frac{w_{11}P_1(k) + w_{21}(P_2(k) - M_{21}) + w_{31}(P_3(k) - M_{31})}{w_{11} + w_{21} + w_{31}} \tag{6.16}$$

$$P_2(k+1) = \frac{w_{12}(P_1(k) - M_{12}) + w_{22}P_2(k) + w_{32}(P_3(k) - M_{32})}{w_{12} + w_{22} + w_{32}} \tag{6.17}$$

$$P_3(k+1) = \frac{w_{13}(P_1(k) - M_{13}) + w_{23}(P_2(k) - M_{23}) + w_{33}P_3(k)}{w_{13} + w_{23} + w_{33}} \tag{6.18}$$

The network has recurrent connections, i.e. for each cycle of simulation, the output values $P_n(k+1)$ are given to the input ($P_n(k) \leftarrow P_n(k+1)$). Furthermore, the network receives, through sensors, the measures of vectors $P_n(k)$ which, in general, do not point to the same position in the plane. This case demonstrates a fundamental property of this network: after an arbitrary input is switched off, the network, after some iterations, relaxes towards a stable state. The most important property of this net is that, after relaxation (say, when $k = \overline{k}$), all output vectors $P_n(\overline{k})$ point to the same position in space even if the input values pointed to different locations.

The speed of the relaxation depends on diagonal weights (the higher the weights w_{nn}, the slower the relaxation). In our calculation we use, as suggested by Cruse [17], $w_{nm} = 1$ if $n \neq m$ and $w_{nm} = 10$ if $n = m$.

When the absolute reference is not available, a safety mechanism acts to control the motion maintaining a correct heading. This is briefly showed in the following. The model uses local and sensory information that correlate the current position of the agent, the target and the perceived environment.

In solving the guidance task, many bio-inspired models have been developed to emulate insect's behavior. The model, proposed in this section, refers to the parameter hypothesis; as said above, the parameter hypothesis assumes that some set of parameters is extracted and stored as the representation for the goal position. In particular:

- it uses simple sensory information (distances and angles);
- it needs at least 3 landmarks in the environment to build an egocentric coordinate system;
- it needs, compared with other models, less computational resources: it's economic and attractive for robotic applications.

The structure is divided into two parts: the first part concerns the memorization of the target position array with respect to landmarks in terms of relative distances and angles; the second part relates to the reaching of the target position. Despite of the RNN model described in section 6.4.2, we operate with vectors, i.e. distances and angles, and not in terms of x and y coordinates. The features of the model are explained in the following. Once the model has estimated the distances between landmarks and target (Fig. 26(a)), the model stores these information in two separate arrays, named $\widehat{x}$ and $\widehat{x}_{target}$ (Fig. 26(a)):

$$\widehat{\mathbf{x}} = \begin{pmatrix} P_1 & P_2 & P_3 \\ \alpha & \beta & \gamma \end{pmatrix}$$

$$\widehat{\mathbf{x}}_{\mathbf{target}} = \begin{pmatrix} S_1 & S_2 & S_3 \\ \alpha_t & \beta_t & \gamma_t \end{pmatrix}$$

During navigation, the model updates the values of the vector $\widehat{x}$ trying to reduce the discrepancy with the $\widehat{x}_{target}$ array. So, the navigation control is actuated through the comparison of the two arrays; in particular, at iteration h, for the landmark j,

the model evaluates the difference between the two arrays in order to decide how to move:

$$step_{hj} = \widehat{X}_i(j) - \widehat{X}_{target}(j) \tag{6.19}$$

where $\widehat{x}_i(j)$ indicates the j^{th} column of the vector $\widehat{x}$ evaluated at the step h. For each iteration, the movement of the agent is determined by the algebraic sum of $step_{hj}$ on all visual cues:

$$step_h = \sum_j step_{hj} \tag{6.20}$$

After moving, the model updates arrays $\widehat{x}$, measuring current distances and angles. The procedure stops when the agent reaches the nest, i.e. $\widehat{x} = \widehat{x}_{target}$. The peculiarity of such algorithm is that it does not need an absolute reference system and uses only visual information. However, as said above, it needs at least three landmarks, even if not contemporary visible by the agent. For a geometrical overview of the model, refer to Fig. 26(a).

6.4.3 Simulation results

The simulations were made in an arena of 200×200 pixels, filled with colored objects. These objects are considered as visual cues, some of which are fixed, while others can move into the environment. In particular, for simulations reported here, the number of visual cues distributed in the environment is fixed at eight: three visual cues have been located near the target and are stable in time and space, other two visual cues have been considered stable but quite far from the target position, while the remaining visual cues have been randomly placed in the arena and are considered as moving objects. The agent is able to detect the presence of the target within a radius of 10 pixels (i.e. low level target detection sensor) while for landmark detection a visibility cone has been defined, centered on the front side of the robot and with a range of $[-45°; +45°]$.

The simulated robot, during the exploration phase, is controlled by a reflexive navigation strategy like the Weak Chaos Control and the movement, in absence of stimuli, is guided by the evolution of the chaotic system. This behaviour permits a complete exploration of the area. In the following subsection we report the simulation results related to the landmark identification, while in subsection 6.4.3.2 we show the results related to the two navigation approach using the landmark previously identified as the most reliable.

6.4.3.1 Landmark identification

At each simulation step, if the target is within the detection range of the agent, the input for the target detector neuron is $I = 30$. In the case of visual cues, if the of the visibility conditions are respected, the input of visual cues detectors is a normalized distance, so that the input of these neurons is $I_j = 50e^{-\gamma d_j}$, where d_j is the distance in pixels between the robot and j-th visual cue. It has been chosen $\gamma = 0.02$, so that the range in which visual cues detectors approximatively begin to fire is approximatively 30 pixels from the corresponding cue. Once the neuron inputs have been calculated, a new simulation step is computed. For each simulation, the model behavior is simulated with a time window $300ms$. Simulation cycles have been fixed at 5000. The simulation environment is represented in Fig. 6.27.

During the phase of landmark identification, the robot explores the environment while the STDP rule updates weight values. The simulation stops after 5000 steps of the agent if at least 3 visual cues have overcome the threshold of 1.5. In Fig. 6.28, the evolution of synaptic connection values during simulation is illustrated. So, the model is able to distinguish correctly which of the visual cues are reliable landmarks and which are not.

6.4.3.2 Landmark navigation

Egocentric system with absolute reference

Once three visual cues have been selected as reliable landmarks, the network of Fig. 26(b) can be used to solve the guidance task. When the simulated robot after a foraging task, needs to return to its home (e.g. for recharge its battery), it looks for a landmark. When one of the reliable landmarks is identified, its position with respect to the absolute reference system created with a compass sensor, is passed to the RNN. This information is used to substitute one of the inputs of the recurrent network. The stimulation is applied for only one step and the net has time to relax for the subsequent $\overline{k} - 1$ steps of relaxation (i.e. no input is given during that time). In our simulations, we let the network evolve for $\overline{k} = 20$ step of relaxation. After the relaxation, the new output $P_n(\overline{k})$ points to a position that is nearer to S compared with the actual position P.

The redundant structure of the system also gives rise to another interesting property of the network. If this landmark, during the simulation, becomes not visible due to, for example, occlusion or noise, another can be identified and selected.

The result of the navigation control, driven by the two RNNs (i.e. one for each cartesian coordinate of the absolute reference system built on the knowledge of the North), is shown in Fig. 6.29 where the trajectory of the robot, during the homing phase is reported.

As said above, the model can work well even if there is a single visible landmark in the environment. The advantage of using more than one landmark, i.e. redundant information, is due to the possibility of filtering some types of noise that can affect the sensory system of the agent. In order to test the model robustness against noise,

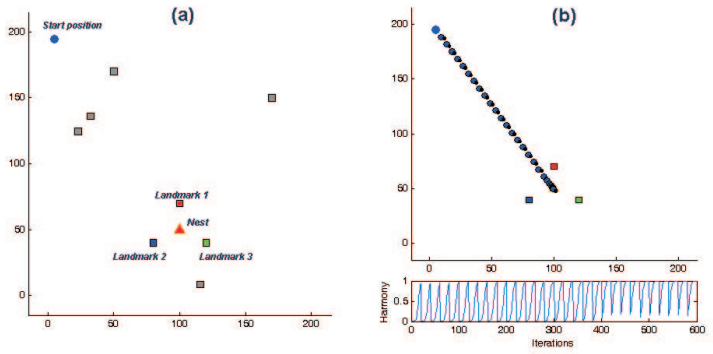


Fig. 6.29 a. Egocentric system without absolute reference: the initial configuration with the three recognized landmarks and the start position of the agent. b. Trajectory followed by the agent: at each time step, one (randomly chosen) landmark, say M_i, is visible and the coordinate of the distance vector P_i set the initial condition $P_i(0)$ for the x-coordinate RNN and for the y-coordinate RNN (see text for details). Harmony value is shown for 600 iteration cycles.

two types of noises have been introduced: noise on distance estimation M_{ij} and S_i, stored in memory, and in the measurement of P_i, acquired at each iteration. In Fig. 6.30, the model results are shown when the sensory system is affected by both types of noise. The model has been tested with increasing values of noise, with a variation on measures due to noises that reach a maximum value of 50% with respect to the correct measures.

The model seems to be able to navigate very well even if there are significant noisy signals acquired by the sensory system.

Egocentric system without absolute reference

When the absolute reference is not available (e.g. either the compass sensor is not equipped on the robot or the environmental conditions don't permit such a measure), the agent can utilize, for navigating in the environment, an alternative navigation model, according to the parameter hypothesis, that makes use of local and sensory information to build an egocentric coordinate system. Fig. 6.31 shows an example of the trajectory of the agent in the case of three landmarks navigation control based on the egocentric model without the absolute reference. In this case, the Harmony index, has been defined an indicator for the relaxation to target, i.e, the sum of square error, which measures the discrepancy between the two arrays $\widehat{X}$ and $\widehat{X}_{target}$.

As in the previous case, the model has been tested with different levels of noises on the sensory system (see Fig. 6.32), demonstrating the navigation ability in this extremely difficult environmental conditions.

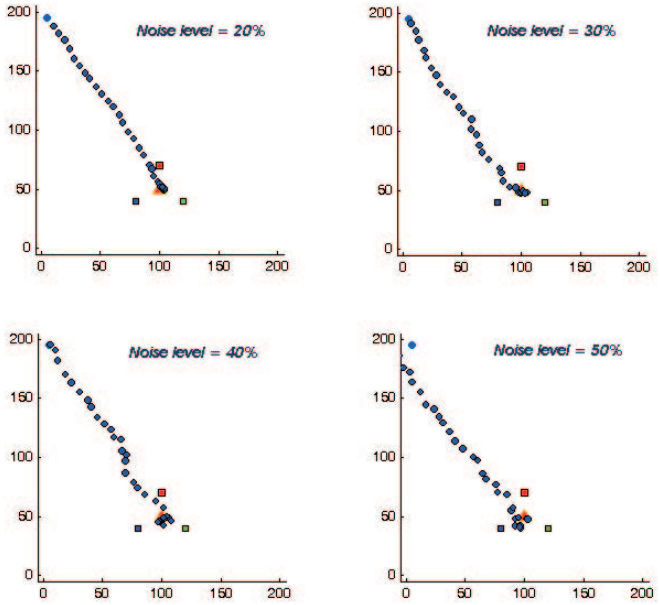


Fig. 6.30 Egocentric system with absolute reference: example of trajectories in presence of different levels of noise in the estimation of M_{ij}, S_i and in the measurement of P_i at each step $(i, j = 1, 2, 3)$.

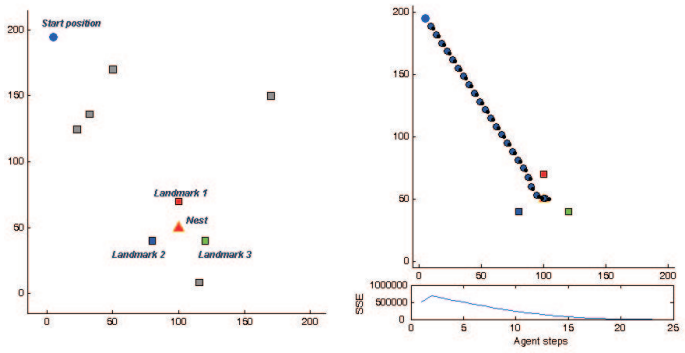


Fig. 6.31 Egocentric system without absolute reference: example of the trajectory using three landmarks. At each time step, all the three landmarks are visible and agent estimates distances and angles with respect to landmarks and moves in order to obtain the equilibrium of reaching the target(see text for details). Dynamic error value is shown for 20 iteration cycles

6.5 Conclusions

In this Chapter we discuss on the basic and correlation layers of the cognitive architecture that will be formalized in Chapter 7.

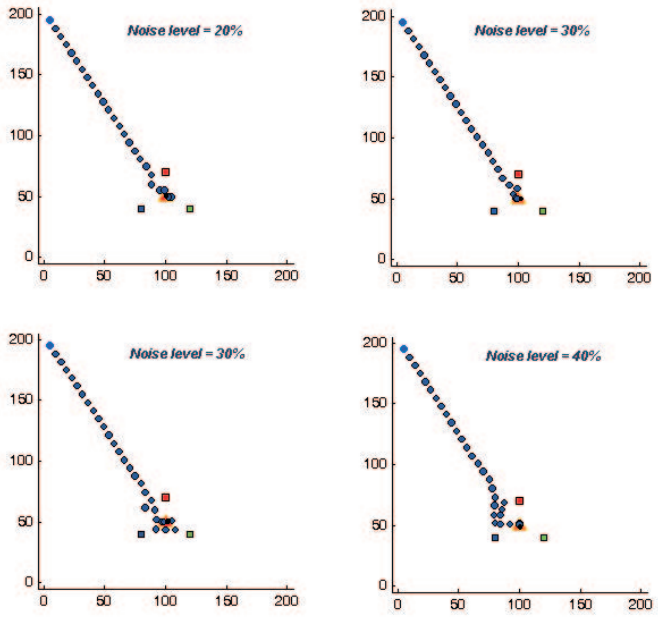


Fig. 6.32 Egocentric system without absolute reference: example of trajectories in presence of different levels of noise in the estimation of M_{ij}, S_i and in the measurement of P_i at each step $(i, j = 1, 2, 3)$.

The discussion is structured into three main blocks. The first one concerns the application of dynamical systems to model reactive and precognitive behaviours. The second one is focused on the correlation mechanisms used to create temporal dependencies among different stimuli and consequently different motor responses. Finally an application to landmark navigation is proposed to fuse together reactive behaviours and anticipation mechanisms.

As presented in the first part of the Chapter, the problem of multi-sensory integration has been treated using a new technique called *weak chaos control*. This approach takes inspiration from the Freeman's theory of brain pattern formation, although it makes use of a more abstract model, and has been applied to a navigation control problem. The phenomenon of encoding information stabilizing the unstable orbits endowed in a chaotic attractor has been investigated. The multiscroll chaotic system was chosen for its simple model and for the possibility to extend the emerging multiscroll attractor to one, two and three dimensions varying also the number of scrolls. The feedback from the environment has been introduced by using a continuous multi-reference chaos control technique based on the Pyragas' method. Extending the discussion for the double scroll to a general *nxm* scrolls configuration. The analytical study has been exploited to develop a reactive navigation

layer for a roving robot. The robot behavior during a food retrieval task has been evaluated in a 3D simulation environment.

In the second part of the Chapter, a network of spiking neurons for robot navigation control is introduced. In particular, by using stimuli of increasing complexity, a system based on spiking neurons and able to implement three local navigation skills (obstacle avoidance, target approaching and navigation with visual cue) has been considered. For all the tasks a priori response to low level sensors (i.e. contact sensors in the case of obstacles, proximity target sensors in the case of target approaching or visual targets in the case of navigation with visual cues) is known and the robot has to learn the response to high level stimuli (i.e. range finder sensors or visual input). The learning mechanism of these networks is provided by STDP, which allows unsupervised learning to be realized in a simple way.

Our approach is a bottom-up approach: we searched for the minimal structure able to implement the desired behaviors. From our analysis we can conclude that a system of spiking neurons with plastic synapses can be used to control a robot and that STDP with additional mechanisms can be used to include plasticity in the system, learning temporal correlation among stimuli.

We believe that the introduced approach can provide efficient navigation control strategies with very simple unsupervised learning mechanisms and at the same time can constitute a constructivist approach which can be interesting for studies on navigation strategies. The approach is constructivist in the sense that the same network structures for the low level tasks are duplicated for dealing with the high level tasks which are based on the acquired (developed) capabilities. Furthermore, this strategy encourages the use of parallel structures which can be included for instance to take into account other visual cues.

Finally we applied the reactive and the correlation layers previously described to formalize a new methodology for adaptive bio-inspired navigation. According to experiments performed with insects, these are capable to use different kinds of information to focalize the nest position, in order to reach it from different distances. The introduced methodology make in evidence how to integrate different simple behaviors and learning algorithms, to obtain more complex pre-cognitive capabilities.

The proposed simulation shows how a simulated robot can distinguish reliable landmarks in an adaptive way for homing porpoises. This means that moving or distant objects are discarded, while fixed ones are kept. This initial setting of the landmark configuration is then applied to a recurrent linear network in order to guide the robot to the nest. The network, although very simple, shows interesting capabilities to filter out noise, even at a relevant level. The robustness of the approach is relevant, also since the reliable working is guaranteed also when the landmarks are not always all visible by the agent. Experiments with roving robots have been carried out to verify the simulation results; details on the hardware experiments are given in Chapter 11.

References

1. Anderson, A.M.: A model for landmark learning in the honey-bee. Journal of Comparative Physiology A **114**(335) (1977)
2. Arena, P., Basile, A., Bucolo, M., Fortuna, L.: An object oriented segmentation on analog CNN chip. IEEE Trans. CAS I **50**(7), pp. 837–846 (2003)
3. 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. International Journal of Bifurcation and Chaos, **18**(1), pp. 107-127 (2007)
4. Arena, P., De Fiore, S., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: Weak Chaos Control for Action-Oriented Perception: Real Time Implementation via FPGA. In Proc. International conference on Biomedical Robotics and Biomechatronics (Biorob), Pisa, Italy, February 20-22, (2006)
5. Arena, P., De Fiore, S., Frasca, M., Patané, L.: Web page available on-line: http://www.scg.dees.unict.it/activities/biorobotics/perception.htm (2006)
6. Arena, P., Fortuna, L., Frasca, M., Lo Turco, G., Patané, L., Russo, R.: A new simulation tool for action oriented perception systems. In Proc. 10th IEEE International Conference on Emerging Technologies and Factory Automation, (ETFA) September 19-22 Catania, Italy (2005)
7. Arena, P., Fortuna, L., Frasca, M., Patané, L., Barbagallo, D., Alessandro, C.: Learning high-level sensors from reflexes via spiking networks in roving robots. In: Proceedings of 8th International IFAC Symposium on Robot Control (SYROCO), Bologna (Italy) (2006)
8. Arkin, R.C.: Behaviour Based Robotics, MIT Press (1998)
9. Beard, R., McClain, T.: Motion Planning Using Potential Fields, BYU (2003)
10. Bingman, V.P., Gagliardo, A., Hough, G.E., Ioalé, P., Kahn, M.C., Siegel, J.J.: The Avian Hippocampus, Homing in Pigeons and the Memory Representation of Large-Scale Space. Integr. Comp. Biol. **45**, pp. 555–564 (2005)
11. Boccaletti, S., Grebogi, C., Lai, Y.C., Mancini, H., Maza, D.: The Control of Chaos: Theory and Applications. Physics Reports, **329**, 103–197 (2000)
12. Burgess, N., Becker, S., King, J.A., O'Keefe, J.: Memory for events and their spatial context: models and experiments. Phil. Trans. R. Soc. Lond. B **356**, pp. 1–11 (2001)
13. Burgess, N., O'Keefe, J.: Neuronal computations underlying the firing of place cells and their role in navigation. Hippocampus **6**, pp. 749–762 (1996)
14. Burgess, N., Recce, M., O'Keefe, J.: A model of hippocampal function. Neural Networks **7**, pp. 1065–1081 (1994)
15. Cartwright, B.A., Collett, T.S.: Landmark learning in bees. The Journal of Comparative Phisiology A **151**(85) (1983)
16. Collett, T.S.: Insect navigation en route to the goal: Multiple strategies for the use of landmarks. The Journal of Experimental Biology **202**, pp.1831–1838 (1999)
17. Cruse, H.: A recurrent network for landmark-based navigation. Biological Cybernetics **88**, pp. 425–437 (2003)
18. Floreano, D., Mattiussi, C.: Evolution of Spiking Neural Controllers for Autonomous Vision-based Robots. Evolutionary Robotics IV, Berlin Springer-Verlag (2001)
19. Franceschini, N., Blanes, C.: From insect vision to robot vision. Philosophical Transaction of the Royal Society of London B **337**, pp. 283–294 (1992)
20. Franz, M.O., Mallot, H.A.: Biomimetic robot navigation. Robotics and Autonomous Systems **30**, pp. 133–153 (2000)
21. Freeman, W.J.: Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. Biol. Cybern., **56**, pp. 139-150 (1987).
22. Freeman, W.J.: The physiology of perception. Sci. Am. **264**, 78–85 (1991)
23. Freeman, W.J.: Characteristics of the Synchronization of Brain Activity Imposed by Finite Conduction Velocities of Axons. International Journal of Bifurcation and Chaos, **10**(10), (1999)

24. Freeman ,W.J.: A Neurobiological Theory of Meaning in Perception. Part I: Information and Meaning in Nonconvergent and Nonlocal Brain Dynamincs. International Journal of Bifurcation and Chaos **13**(9) (2003)
25. Freeman, W.J.: How and Why Brains Create Meaning from Sensory Information. International Journal of Bifurcation and Chaos, **14**(2) (2004)
26. Fuster, J.M.: Cortex and Mind: Unifying Cognition, Oxford University Press (2003)
27. Grossberg, S., Maass, W., Markram, H.: Introduction: Spiking Neurons in Neuroscience and Technology. Neural Networks, special issue on Spiking Neurons **14**(6-7), p. 587, (2001)
28. Harter, D.: Evolving neurodynamics controllers for autonomous robots. In: International Joint Conference on Neural Networks, pp. 137–142 (2005)
29. Harter, D., Kozma, R.,: Chaotic Neurodynamics for autonomous agents. IEEE Trans. on Neural Networks, **16**(3), pp. 565–579 (2005)
30. Izhikevich, E.M.: Simple Model of Spiking Neurons. IEEE Transactions on Neural Networks **14**(6), pp. 1569–1572 (2003)
31. Izhikevich, E.M.: Which Model to Use for Cortical Spiking Neurons?. IEEE Transactions on Neural Networks **15**(5), pp. 1063–1070 (2004)
32. Izhikevich, E.M.: Solving the distal reward problem through linkage of STDP and dopamine signaling. Cerebral Cortex Advance (2007)
33. Izhikevich, E.M., Gally, J.A., Edelman, G.M.: Spike-Timing Dynamics of Neuronal Groups. Cerebral Cortex **14**, pp. 933–944 (2004)
34. Jensen, O., Lisman, J.E.: Hippocampal sequence-encoding driven by a cortical multi-item working memory buffer. TRENDS in Neurosciences, **28**(2) (2005)
35. Khatib, O.: Real-time Obstacle Avoidance for Manipulators and Mobile Robots. Intemational Journal of Robotics Research **5**(1), pp 90–98 (1986)
36. Koene, R.A., Gorchetchnikov, A., Cannon, R.C., Hasselmo, M.E.: Modeling goal-directed spatial navigation in the rat based on physiological data from the hippocampal formation. Neural Networks **16**, pp. 577–584 (2003)
37. Kozma, R., Freeman, W.J.: Chaotic Resonance - Methods and Applications for Robust Classification of Noisy and Variable Patterns. International Journal of Bifurcation and Chaos **11**(6) (2000)
38. L*ü*, J., Chen, G., Yu, X., Leung, H.: Design and Analysis of Multiscroll Chaotic Attractors from Saturated Function Series. IEEE Trans. Circuits Syst., I: Regular Paper **51** (2004)
39. Manganaro, G., Arena, P., Fortuna, L.: Cellular Neural Networks: Chaos, Complexity, and VLSI Processing. Springer-Verlag, Berlin (1999)
40. Nicholson, D.J., Judd, S.P.D., Cartwright, A., Collett, T.S.: View-based navigation in insects: how wood ants (Formica rufa L). The Journal of Experimental Biology **202**, pp. 1831–1838 (1999)
41. Pavlov, I.P.: Conditioned Reflexes. Oxford University Press, London (1927)
42. Pyragas, K.: Continuos Control of Chaos by Self-Controlling Feedback. Phisical Letters A **170**, pp. 421–428 (1992)
43. Pyragas, K.: "Predictable Chaos in Slightly Pertirbed Unpredictable Chaotic Systems. Physics Letters A **181**, pp.203–210 (1993)
44. Restle, F.: Discrimination of cues in mazes: A resolution of the 'place-vs-response' question. Psychological Review **64**(4), pp. 217–228 (1957)
45. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-Organizing Maps - An Introduction. Addison-Wesley, New York (1992)
46. Shepherd, G.M: Neurobiology. Oxford University Press (1994)
47. Skarda, C.A., Freeman, W.J.: How brains make chaos in order to make sense of the world. Behav. Brain Sci. **10**, pp. 161-195 (1987)
48. Song, S., Abbott, L.F.: Cortical development and remapping through Spike Timing-Dependent Plasticity. Neuron **32**, pp. 339–350 (2001)
49. Song, S.,Miller, K.D., Abbott, L.F.: Competitive Hebbian learning through spike-timing-dependent plasticity. Nature Neurosci. **3**, pp. 919–926 (2000)
50. Trullier, O., Wiener, S.I., Berthoz, A., Meyer, J.A.: Biologically-based Artificial Navigation Systems: Review and prospects. Progress in Neurobiology **51**, pp. 483–544, (1997)

51. Uexku, J.V.: Theoretical Biology. Harcourt, Brace (1926)
52. Verschure, P.F.M.J., Kröse, B.J.A., Pfeifer, R.: Distributed adaptive control: The self-organization of structured behavior. Robotics and Autonomous Systems **9**, pp. 181–196 (1992)
53. Verschure, P.F.M.J., Pfeifer, R.: Categorization, Representations, and the Dynamics of System-Environment Interaction: a case study in autonomous systems. In J.A. Meyer, H. Roitblat, S. Wilson (Eds.) From Animals to Animats: Proceedings of the Second International Conference on Simulation of Adaptive Behavior. Cambridge, MA, MIT Press, pp. 210–217 (1992)
54. Webb, B., Consi, T. R.: Biorobotics. MIT Press (2001)
55. Webb, B., Scutt, T.: A simple latency dependent spiking neuron model of cricket phonotaxis. Biological Cybernetics **82**(3), pp. 247–269 (2000)
56. Weber, A.K., Venkatesh, S., Srinivasan, M.: Insect-inspired robotic homing. Adaptative behavior **7**, pp.65–97 (1999)
57. Wehner, R., Michel, B., Antonsen, P.: Visual navigation in insects: Coupling of egocentric and geocentric. The Journal of Experimental Biology **199**, pp. 129–140 (1996)
58. Zampoglou, M., Szenher, M., Webb, B.: Adaptation of Controllers for Image-Based Homing. Adaptive Behaviour **14**(4), pp.381–399 (2006)