Learning Anticipation via Spiking Networks: Application to Navigation Control

Paolo Arena, *Senior Member, IEEE*, Luigi Fortuna, *Fellow, IEEE*, Mattia Frasca, *Member, IEEE*, and Luca Patané, *Student Member, IEEE*

***Abstract*—In this paper, we introduce a network of spiking neurons devoted to navigation control. Three different examples, dealing with stimuli of increasing complexity, are investigated. In the first one, obstacle avoidance in a simulated robot is achieved through a network of spiking neurons. In the second example, a second layer is designed aiming to provide the robot with a target approaching system, making it able to move towards visual targets. Finally, a network of spiking neurons for navigation based on visual cues is introduced. In all cases, the robot was assumed to rely on some *a priori* known 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). Based on their knowledge, the robot 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) 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.**

***Index Terms*—Navigation control, spiking neurons, spike-timing-dependent plasticity (STDP).**

I. Introduction

According to the definition discussed in [1], navigation skills can be divided in *local navigation* skills and *way-finding* skills. Local navigation skills are based on information on the immediate environment perceived by the robot, and thus, opposite to way finding, they do not necessarily require an internal representation of the surrounding world. In robotics, the spectrum of possible behaviors is broad, ranging with continuity from reactive to deliberative processes [2]. The tasks investigated in this paper were implemented by a reactive system augmented with systems for learning, showing that local navigation capabilities can be improved by involving stimuli of increasing complexity.

Manuscript received March 02, 2007; revised September 25, 2007 and June 11, 2008; accepted July 15, 2008. First published January 13, 2009; current version published February 06, 2009. This work was supported by the European Commission under Projects FP6-2003-IST2-004690 SPARK and FP7-ICT-2007-1-216227 SPARK II.

The authors are with the Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi, Università degli Studi di Catania, 95125 Catania, Italy (e-mail: parena@diees.unict.it; lfortuna@diees.unict.it; mfrasca@diees.unict.it; lpatane@diees.unict.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2005134

Although the tasks considered are basic problems in navigation control and have been solved using many different approaches [2], our idea is to investigate a suitable navigation control scheme, which relies on learning strategies in spiking neurons [3] (i.e., the fundamental processing units of the nervous system).

Literature on navigation control as well as the papers focusing on biologically inspired approaches to navigation are vast. A review on bioinspired navigation control schemes is provided by Trullier *et al.* [1] and by Franz and Mallot [4]. A key role in navigation control in animals is played by hippocampal cells, which encode spatial locations and act as an associative memory [5]–[9]. The aim of our paper is not to reproduce a biological model, but to use the paradigm of spiking computation to build an artificial structure suitable for controlling a roving robot.

Among bioinspired approaches to navigation, the work by Floreano *et al.* [10], [11] is worth mentioning. Here spiking neural circuits are applied to control navigation in a miniaturized roving robot. Floreano *et al.* use a spike response model to build a network of spiking neurons controlling the robot. They genetically encode and evolve the pattern of connectivity among neurons and between neurons and sensory receptors. Their approach is thus based on the evolution of circuit parameters rather than on their adaptation during conditional learning.

A very interesting approach is the distributed adaptive control model, based on artificial neural networks introduced in [12] and [13]. Another important difference is that in our strategy the response to a given stimulus is modulated by the stimulus itself, while in the model discussed in [12] and [13] responses are predefined, being selected by the stimulus through a set of command neurons.

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 (a key issue of the navigation control that the system has to include) should be based on a biologically plausible mechanism. Several hypotheses have been formulated to model the biological mechanisms of plasticity. It has been recently pointed out that a primary role is played by spike timing [14], [15]. Consequently, the so-called spike-timing-dependent plasticity (STDP) has been introduced to explain biological plasticity [14], [15]. According to this 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 presynaptic spike occurs before the postsynaptic 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. In our system, successive repetitive occurrences of the stimulus lead to reinforce the conditioned response, according to the paradigms of classical and operant conditioning [16], [17], 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 [17]. 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 [17]. These two biological mechanisms, used by living creatures to learn from 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 US, while range finder sensors represent 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 bioinspired learning approach is potentially relevant to the goal of progressively including, into the navigation control loop, more and more details gained from high-level sensors, such as visual cues. As learning proceeds, new visual features could be used to select more appropriate actions, relevant to the current task.

STDP networks are widely studied: it has been demonstrated that they are able to generate map structures [18], [19], to explain the emergence of direction selectivity [20], to represent temporal receptive fields [21], to store patterns [22], and to study the dynamics of cortical neurons [15]. Many of these papers are devoted to study network properties that are related to important features of the brain (for instance, cortical development [18] or map formation in the Barn Owl's brain [19]), and to this aim make the assumption that the input is a random variable, characterized by given statistical properties. In our paper, instead, 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.

The novel aspects of our work concern three points: conditioning, spiking neurons, and the use of STDP in robot controllers. With respect to conditioning, we explicitly formulate a model in which learning occurs via a loop through the world. A neuron model (proposed by Izhikevich [23]), which explicitly takes into account the dynamics underlying spike generation and frequency coding of the input intensity, is used. A learning rule based on STDP with additional mechanisms is applied to an example of navigation control in a roving robot. Our simulations, in fact, led us to conclude that STDP needs further assumptions to properly work for robot control. STDP with clipping and an additional decay rate of synaptic weights is shown to be suitable for simple navigation tasks, whereas in the example of navigation with visual cues, the STDP rule does not work at biologically plausible delays and some other mechanisms must account for this.

The paper is organized as follows. The network model is introduced in Section II and the simulation environment in Section III. Obstacle avoidance is illustrated in Section IV. Section V discusses the network for target approaching and the related simulation results. In Section VI, navigation based on visual cues is discussed. The conclusions of the paper are drawn in Section VII.

II. 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. The network consists of interacting spiking neurons, which may play the role of sensory neurons, interneurons, or motor neurons. Sensory neurons are neurons connected to sensors, while motor neurons drive robot motors; interneurons are all the other neurons involved in the processing for navigation control. Each neuron is modeled by the following equations proposed by Izhikevich [24]:

$$\begin{aligned} \dot{v} &= 0.04v^2 + 5v + 140 - u + I \\ \dot{u} &= a(bv - u) \end{aligned} \tag{1}$$

with the spike resetting

$$\text{if } v \geq 0.03, \qquad \text{then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2}$$

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 millisecond.

According to the parameters chosen [23], 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, we focussed on class I excitable neurons. In these neurons, the spiking rate is proportional to the amplitude of the stimulus [23]. 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, which are thus class I excitable neurons able to encode the distance from the obstacle through their firing rate, as schematically shown in Fig. 1. Neuron parameters are chosen as $a = 0.02, b = -0.1, c = -55, d = 6$ (class I

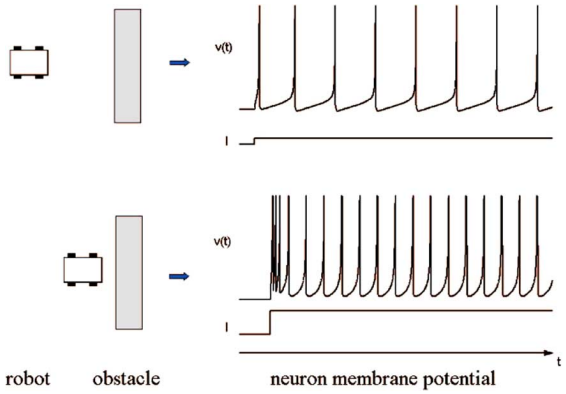


Fig. 1. Class I excitable neurons encode the robot distance from the obstacle into their firing frequency.

excitable neurons [23]), 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.

Concerning the model of the synapse, let us consider a neuron j, which receives synaptic connections from 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:

$$I_j(t) = \sum w_{ij}\varepsilon(t - t_i) \tag{3}$$

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:

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau}e^{1-\frac{t}{\tau}}, & \text{if } t \geq 0 \\ 0, & \text{if } t < 0. \end{cases} \tag{4}$$

Equation (4) describes the contribution of a spike, from a presynaptic neuron emitted at $t = 0$ [10]. In our simulations, τ has been fixed to $\tau = 5$ ms.

To include adaptive capabilities in our model, Hebbian learning was considered. Recent results [14] 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 [14] and here briefly reported. Let us indicate with w the synaptic weight. A presynaptic spike and a postsynaptic spike modify the synaptic weight w by $w \rightarrow w + \Delta w$, where, according to the STDP rule, Δw depends on the timing of presynaptic and postsynaptic spikes. The following rule holds [14]:

$$\Delta w = \begin{cases} A_+ e^{\frac{\Delta t}{\tau_+}}, & \text{if } \Delta t < 0 \\ A_- e^{\frac{-\Delta t}{\tau_-}}, & \text{if } \Delta t \geq 0 \end{cases} \tag{5}$$

where $\Delta t = t_{\text{pre}} - t_{\text{post}}$ is the difference between the spiking time of the presynaptic neuron (t_{pre}) and that of the postsynaptic one (t_{post}). If $\Delta t < 0$, the postsynaptic spike occurs after the presynaptic spike of the synapsis is reinforced. Otherwise, if $\Delta t \geq 0$, the postsynaptic spike occurs before the presynaptic spike and 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 (5) is a rather standard assumption to model STDP. The term $A_+(A_-)$ represents the maximum Δw, which is obtained for almost equal pre- and postspiking times in the case of potentiation (depression).

The use of the synaptic rule described by (4) may lead to an unrealistic growth of the synaptic weights. For this reason, often upper limits for the weight values are fixed [18], [25]. Furthermore, some authors (see, for instance, [12] and [26]) introduce a decay rate in the weight update rule. This solution avoids that the weights of the network increase steadily with training and allows for a continuous learning to be implemented. In our 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:

$$w(t+1) = \begin{cases} 0.95w(t) + \Delta w, & \text{if } t \bmod 3000 = 0 \\ w(t) + \Delta w, & \text{otherwise} \end{cases} \tag{6}$$

where $t \bmod 3000$ indicates the modulus of t divided by 3000 and Δw is given by (5).

In the following, the upper limit for 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.

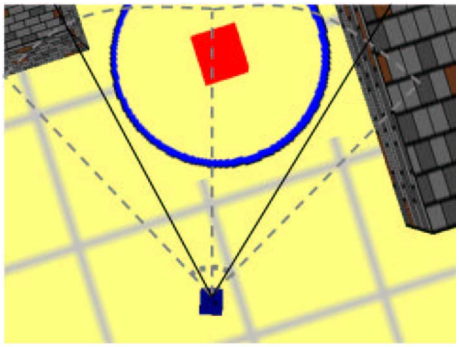
III. Robot Simulation and Controller Structure

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

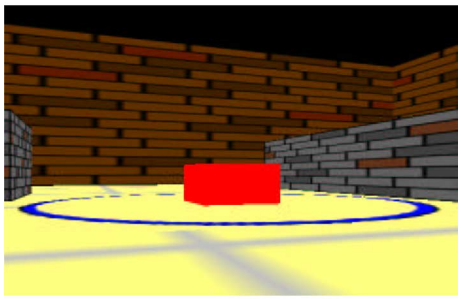
The simulated robot has a cubic shape: in the following, the edge of the robot will be used as measure unit, and indicated as robot unit (r.u.). The simulated robot is a dual-drive wheeled robot: the two motors are labeled as left (LM) and right motor (RM). The two actuated wheels are driven by the output of the spiking network 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°, +45°] (right sensor) or [−45°, 0°] (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. 2: a schematic representation of sensor ranges is given in Fig. 2(a), while in Fig. 2(b) the onboard camera view is shown.

The simulation environment is written in C++ and the visualization is based on the OpenGL libraries [44]. The software has been designed as an environment for the simulations of roving robots controlled by routines easily defined by the user. It allows simulations in a 3-D space. More details on the simulation environment are given in [27]. The 3-D visualization is only needed for the task of navigation with visual cues.

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 interneurons, 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



(a)



(b)

Fig. 2. (a) The simulated robot is equipped with range finder sensors centered on the front side of the robot and with a range of $[0°, +45°]$ (right sensor) or $[-45°, 0°]$ (left sensor), schematically shown with gray dashed lines (outer arcs). The inner arcs refer to the range of contact sensors. The range of the onboard camera view is shown with black solid lines. The circle surrounding the target represents the range in which the target is sensed by the proximity target sensors of the robot. (b) Onboard camera view.

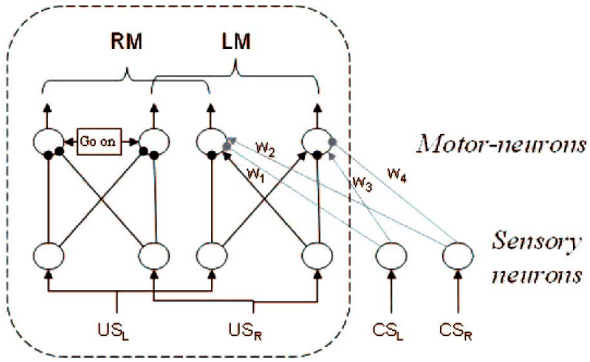


Fig. 3. Network for obstacle avoidance. RM and LM are right and left motors, respectively. US_R and US_L represent right and left contact sensors, respectively, while CS_R and CS_L represent conditioned stimuli (i.e., range finder sensors). Each circle indicates a class I excitable neuron modeled by (1). 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).

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 (1). Let us first 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, and consequently, the robot avoids 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 approximately 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 (388×252 pixels) is preprocessed 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 cases, the filtered image is tiled in nine 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 $p > 400$.

Interneurons constitute an intermediate layer between sensory neurons and motor neurons. We found that this layer is not strictly necessary for 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 constituted by four motor neurons. For the sake of uniformity, these neurons (and the interneurons as well) have been modeled by 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 aspects of the simulation.

The robot moves in a simulated environment filled with randomly placed obstacles. At each simulation step, a time window of 300 ms of model behavior is simulated. The robot moves according to the number of spikes generated by the motor neurons during this time window. A 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 (1) of the associated motor neurons. Referring to the spiking network shown in the box of Fig. 3, 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 spikes 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 0.14 rad (about 8°). In this case, after rotating, the robot proceeds forward. We count the spikes emitted both by the left neuron 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.

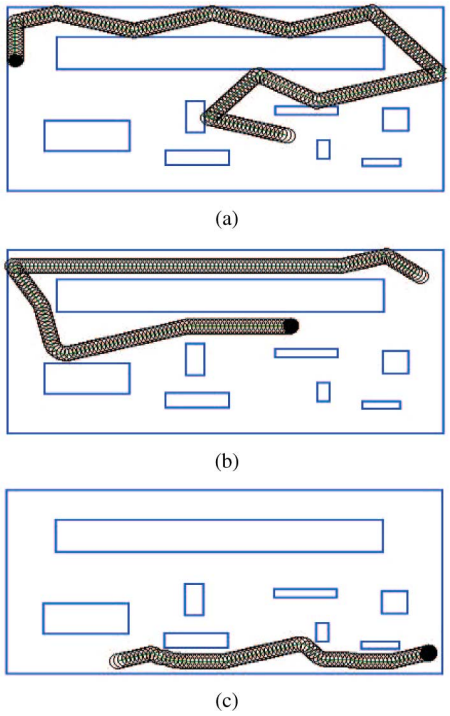


Fig. 4. Three meaningful parts of a simulation devoted to learn CRs. (a) Robot behavior is driven by UR. (b) The robot has already learned CR to left obstacles. After two collisions with right obstacles, it avoids a right obstacle with a CR. (c) The robot avoids obstacles using CR. The final position, for each of the simulation phases, is indicated with a filled circle.

IV. Obstacle Avoidance

The spiking network described in this section deals with the navigation task of moving in 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. 3. It is functionally divided in two parts. The first group of neurons (included in the gray 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 inspired by Braitenberg vehicles [28]. These are simple reactive systems endowed with a bioinspired control based on direct coupling between sensors and motors. In particular, a model with direct inhibition and cross excitation of sensory and motor neurons has been adopted. This model is biologically relevant: in fact, it has been used for modeling cricket phonotaxis in [29]. Concerning sensory neurons, we assume that 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. 3, deals with conditioned stimuli. The synaptic weights of this network evolve according to the STDP rule discussed above. Those ones connecting motor neurons to CS_L and CS_R are not known *a priori* and should be learned during the exploration. The following parameters have been used for the STDP rule: $A_+ = 0.02, A_- = -0.02, \tau_+ = 20$ ms, $\tau_- = 10$ ms.

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 is 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. 4. During the first phase, the robot avoids obstacles by using UR, triggered by contact sensors [Fig. 4(a)]. During this phase, the robot learns the correct CR. Proceeding further with the simulation [Fig. 4(b)], obstacles are finally avoided only by using range finder sensors, i.e., CS [Fig. 4(c)]. The weights converge towards an equilibrium value given by the balance between new

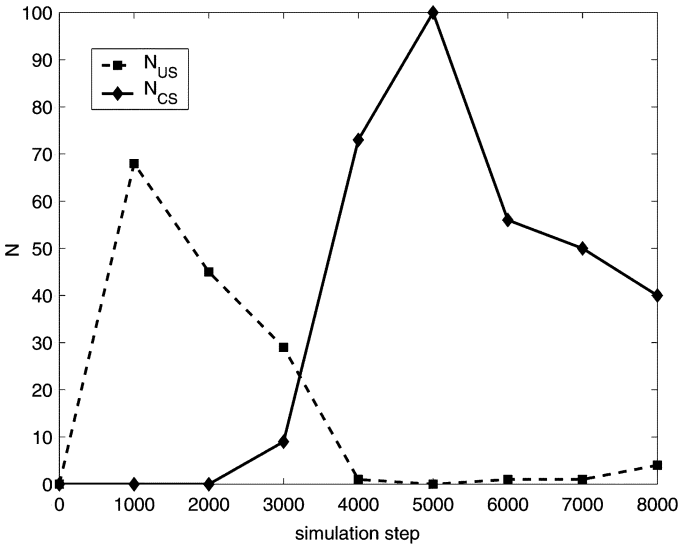


Fig. 5. $N_{\rm CS}$ compared with $N_{\rm US}$ in the case of obstacle avoidance. The time window in which $N_{\rm US}$ and $N_{\rm CS}$ are calculated is 1000 simulation steps.

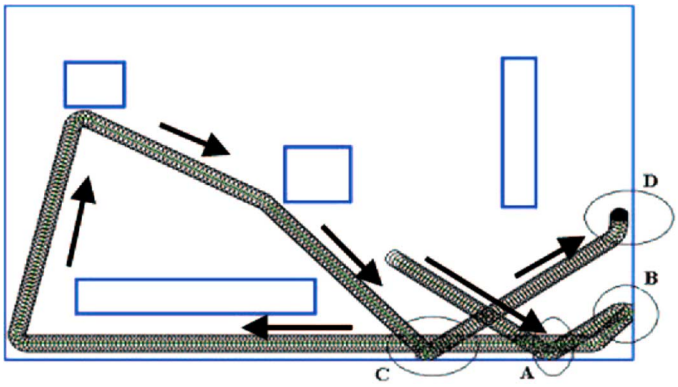


Fig. 6. Robot controlled by the spiking network for obstacle avoidance is able to reorganize its turning strategy. The final point of the trajectory followed by the robot is indicated with a filled circle.

experience, learned when collision sensors are activated, and the weight decay rate.

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. 5, the number of avoidance movements due to CS (namely, $N_{\rm CS}$) is compared with the number of avoidance movements due to US (namely, $N_{\rm US}$). The time window in which $N_{\rm US}$ and $N_{\rm CS}$ 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, 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 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.

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. 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 was 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).

A. *Obstacle Avoidance: Simulation Results*

To evaluate the performance of the spiking network controller, a systematic analysis was carried on. Five different arenas with randomly placed obstacles were considered. For each arena, five different simulations were performed. In each simulation, the robot is placed in a random initial position, which determines a different evolution of the learning algorithm. Thus, a total of 25 different cases is taken into account. The total length of each simulation is $T = 25\,000$ steps. The size of the arenas is 75×75 (in robot units); each arena contains five randomly placed obstacles, which are 10×10 (in robot units) squares.

The behavior of the network controller is evaluated by taking into account two parameters: $\langle N_{\rm US} \rangle$ and $\langle d \rangle$. $\langle N_{\rm US} \rangle$ is defined as the number of collisions $N_{\rm US}$ averaged over the 25 simulations. Then, we define d as the mean distance between the robot and the obstacles, and indicate its average over 25 simulations as $\langle d \rangle$. To calculate d, a temporal mean (over a given time window of 1000 simulation steps) of the distance between robot and closest obstacle is considered.

The aim of the network is to learn how to use CS. Therefore, we expect that, after learning, $\langle N_{\rm US} \rangle$ decreases and $\langle d \rangle$ increases. In fact, after learning, we expect that the number of collisions with obstacles decreases and the robot avoids obstacles at a larger mean distance.

In Fig. 7, $\langle N_{\rm US} \rangle$ is shown as function of the simulation time. As expected, while learning proceeds, the mean number of collisions decreases. Fig. 8 shows $\langle d \rangle$ as function of the simulation

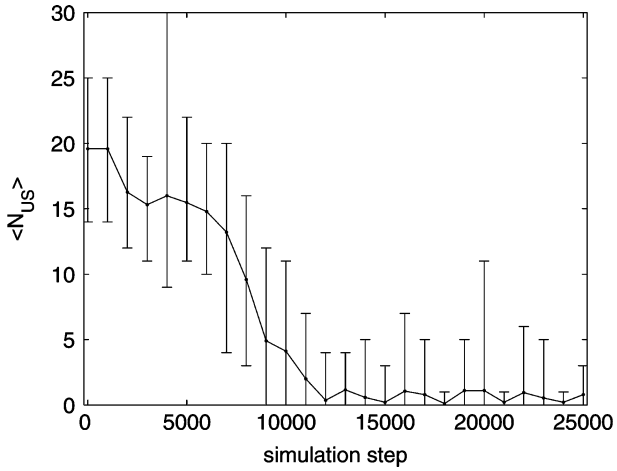


Fig. 7. $\langle N_{\rm US} \rangle$ as function of the simulation time. While learning proceeds, the mean number of collisions decreases. Results are averaged over 25 trials, which take into account five different arenas and five different robot initial positions for each arena. Error bars represent the minimum and maximum values obtained.

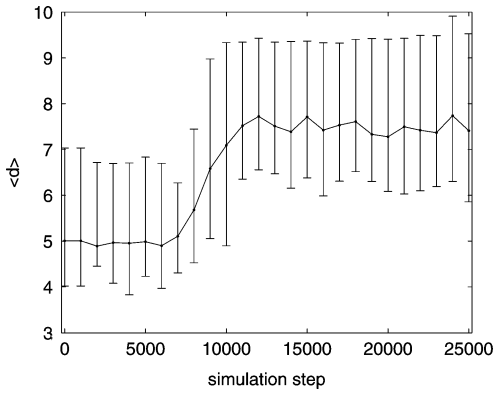


Fig. 8. $\langle d \rangle$ as function of the simulation time. While learning proceeds, the mean distance between robot and obstacles increases. Results are averaged over 25 trials, which take into account five different arenas and five different robot initial positions for each arena. Error bars represent the minimum and maximum values obtained.

time. $\langle d \rangle$ increases as learning proceeds. At the end of the simulation, $\langle d \rangle$ reaches the value of $\langle d \rangle \simeq 8$ r.u., which, compared to the distance at which sensory neurons begin to sense the presence of an obstacle (i.e., about 11 r.u.), demonstrates the efficacy of the controller.

An important factor to show the stability of the algorithm is the trend of the synaptic weights. In Fig. 9, three typical examples are shown. The examples refer to three of the 25 simulations under examination, and report the values of the weights associated to the plastic synapses shown in Fig. 3. In most cases, the trend is similar to that one shown in Fig. 9(a) or (b), where after a transient phase, the weights oscillate around steady-state mean values. Furthermore, two other points showing the suitability of the approach are the stability of robot behavior in long run trajectories (some examples are shown in Section V-B) and the good performance of the approach when applied to a real robot (as shown in Section IV-B). With respect to this point, it should be mentioned that several papers have addressed in detail the stability issue of learning rules exploiting timing of action potentials [30]–[32].

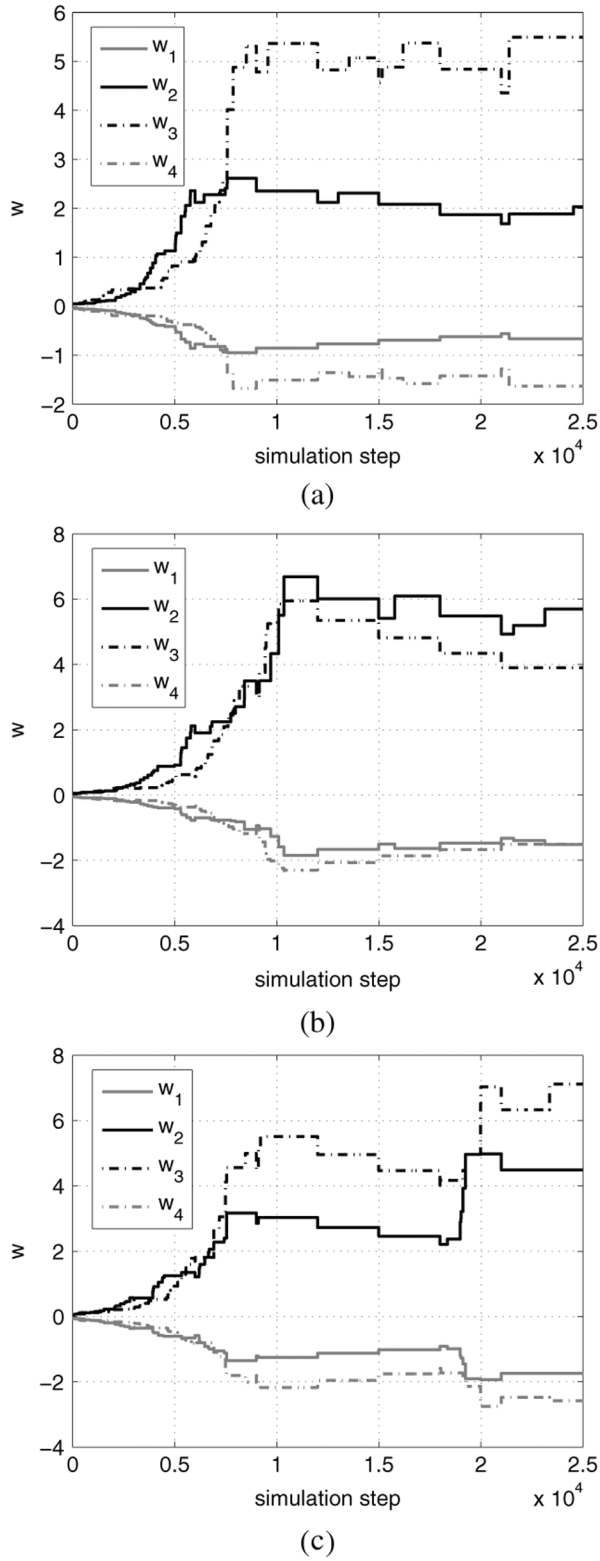


Fig. 9. Trends of the synaptic weights in two typical cases (a) and (b) and in a case (c) in which the robot was engaged into a dead end.

As a consequence of the different values for the weights the robot has a tendency to turn in one direction (i.e., left) more often than in the other direction. As observed above, this is an advantage in critical situations. Fig. 9(c) shows an atypical trend obtained only in one of the 25 simulations analyzed. The

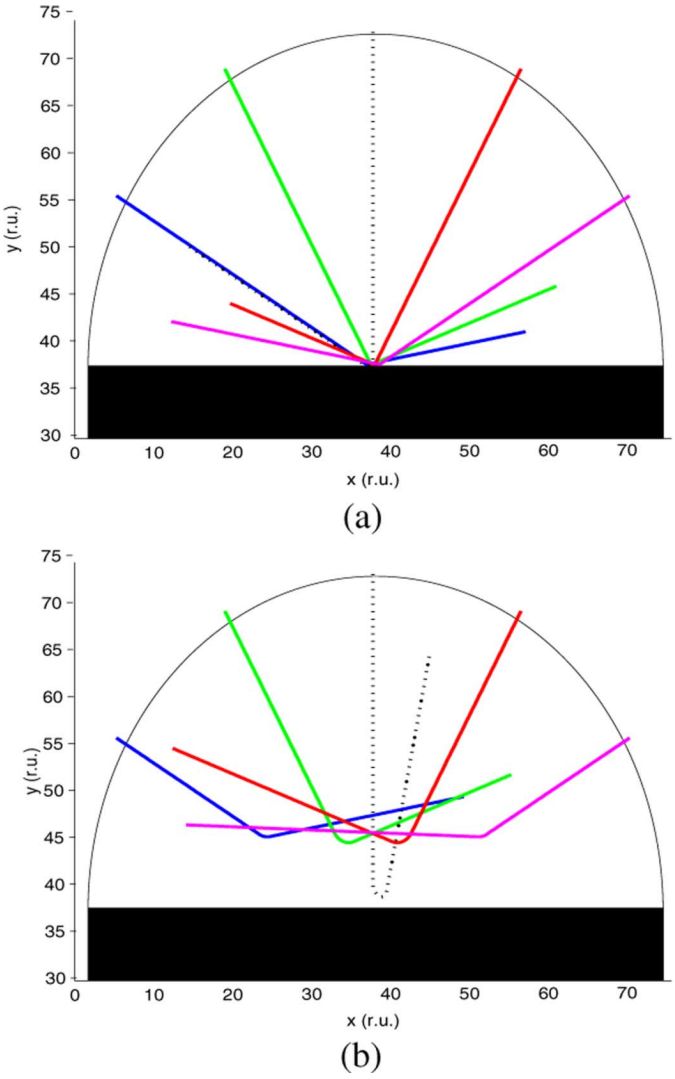


Fig. 10. Trajectories obtained in the case in which the robot faces one obstacle starting at different points equidistant to it: (a) before learning; (b) at the end of the learning phase.

Fig. 11. Behavior of the robot after (a) 3 min and (b) 24 min from the beginning of the learning phase.

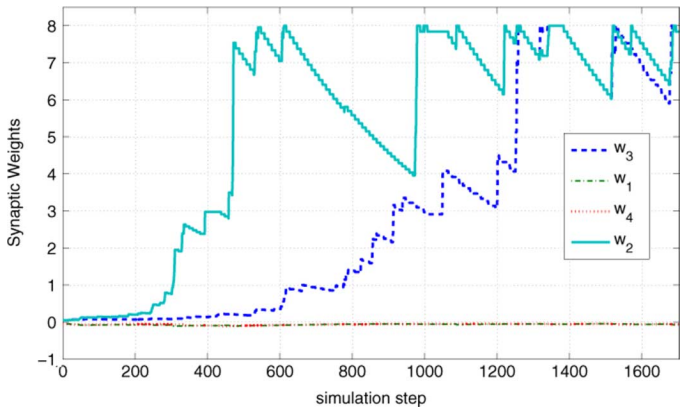


Fig. 12. Robot experiments. Temporal evolution of the synaptic weight connecting distance sensors and interneurons (collision avoidance network). It can be noticed that $w_1 \simeq w_4 \simeq 0$.

weights show a sudden change at $t \simeq 18\,000$ simulation steps. Corresponding to this, N_{US} (not shown) suddenly increases, demonstrating that the robot was engaged in a difficult maneuver (in this case, a corner of the arena). As the trend of the weights after this dead end demonstrates, the robot has successfully dealt with this difficult maneuver.

Another interesting case to evaluate robot performance is when there is only one obstacle (e.g., a wall) and the robot moves towards it starting from different points at the same distance from the obstacle. We evaluated the performance of the robot before, during, and after learning, which was performed in one of the five arenas under examination. The trends of the weights obtained in this case are shown in Fig. 9(b). Before learning [as shown in Fig. 10(a)], the robot starts turning after a collision with the obstacle. During learning, the robot starts turning at an increasing distance to the obstacle. After the learning phase, the robot turns at a distance $d \simeq 9$ r.u. In the case of front obstacle, the robot turns left, since $w_2 > w_3$. Since the turning rate depends on the difference between the number of spikes in left and right neurons (and this difference is very small in case of front obstacles), in case of front obstacles, the robot turns at a smaller distance with respect to the case of lateral obstacles.

B. Robot Experiments

The obstacle avoidance network has been applied to control a real roving robot. The experimental platform is constituted by a computer used only to store data coming from the robot, a field-programmable gate array (FPGA)-based board, hosting an Altera Stratix II (where the control architecture and learning algorithms are implemented), a mobile robot, and a wireless communication system. The robot used for the experimental setup is a Lynxmotion 4WD2 Robot. It is a classic four wheel rover controlled through a differential drive system. The robot dimensions are 35 cm × 35 cm. The roving robot is equipped with four infrared distance sensors GP2D12 from Sharp and with an RF04 USB radio telemetry module for remote control operations and sensory acquisition.

The detection range of the infrared sensors is set from 14 to 80 cm. In this experimental setup, contact sensors are emulated with distance sensors by introducing a threshold at 23 cm that is the minimum distance needed by the robot to escape from an obstacle.

The hardware control loop includes different processing stages. A sensory acquisition stage is performed on the rover by means of a microcontroller, subsequently the sensory information are transferred through a wireless communication module to the FPGA board in which the spiking network has been implemented. The network and the learning algorithms have

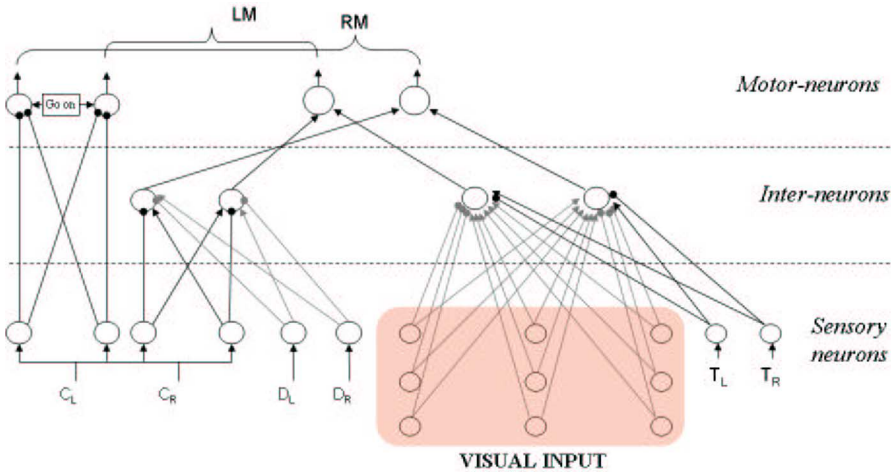


Fig. 13. Network for target approaching based on visual input. D_L, D_R, C_L, and C_R indicate distance/contact left/right sensors. They have been renominated with respect to Fig. 3, in which they were indicated as $\mathrm{CS}_L, \mathrm{CS}_R, \mathrm{US}_L$, and US_R, because now unconditioned stimuli are represented by proximity target sensors (T_L and T_R) and the conditioned stimulus is the visual input.

been developed in the Nios II soft-embedded microprocessor that can be synthesized within the Altera FPGA.

The robot experiment was carried out in an arena (2.5 m $\times$ 2.5 m) filled with randomly distributed obstacles, and the learning session goes on for about 35 min. After learning the robot was able to change its behavior. This is shown in Fig. 11, where the trajectories followed in similar environmental conditions at the beginning and at the end of the learning session are given. Videos about the experiment are available in [33].

The trends of the synaptic weights during learning in this experiment are shown in Fig. 12. The upper saturation was fixed to eight and the decay rate was activated after 40 simulation steps and acts decrementing the synaptic weights of 2% every ten simulation steps. These slightly different parameters have been chosen to speed up the learning phase of the real experiment at the expense of greater oscillations of the weight values. This experiment demonstrates the suitability of the approach in a quite large range of parameters.

V. Target Approaching

A. The Network for Target Approaching

In the previous section, the capability to interpret, in an unsupervised way, high-level sensor stimuli, guided from low-level inherited reflexes, was shown. In this section, the problem of interpreting a visual stimulus is dealt with.

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 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 nine 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 nine compartments, which define the receptive field of each neuron.

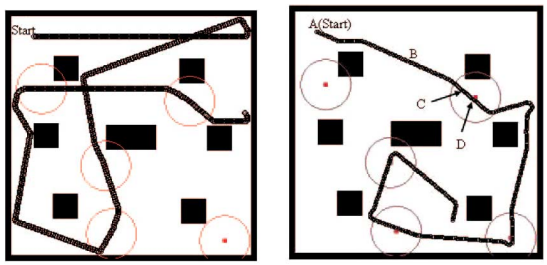
The network of spiking neurons, including visual target approaching, is shown in Fig. 13. Unconditioned stimuli for target approaching are indicated as T_L and T_R (left and right): they are range finder sensors, which detect the distance from the target if the robot is within the proximity range outlined in Fig. 2(a). The whole network is now divided in three layers: sensory neurons, interneurons, and motor neurons. In addition to the plastic synapses of the obstacle avoidance layer, the weights of the synapses between sensory visual neurons and interneurons 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 [34].

B. Target Approaching: Simulation Results

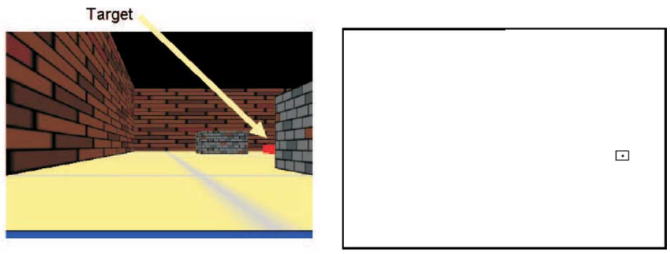
In this section, simulation results focusing on target approaching are shown. We describe several simulations carried out to test the behavior of the robot in a generic environment with randomly placed obstacles and in specific cases, when, for instance, there is a target difficult to find. In any case, 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. 14. 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. 14(a). After learning, the robot follows the trajectory shown in Fig. 14(b).



(a) (b)

Fig. 14. Trajectory followed by the robot with visual target approaching control: (a) without learning; (b) after learning. Targets are indicated with small rectangles inside a circle representing their visibility range.



(a) (b)

Fig. 15. (a) Robot camera view at point A of Fig. 14(b). (b) Visual input: the small rectangle indicates that a target has been identified.

In the case of Fig. 14(b) instead of proceeding in the forward direction as in Fig. 14(a), at point A, the robot rotates since it sees the target that it will reach at point D. This is shown in Fig. 15, 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. 14(a). Fig. 16 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. 17, 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, in this case, on the long run, the robot has learned a trajectory, well tailored on the learning environment, to visit all the targets.

To further test the behavior of the network controller for target approaching, a simulation environment (shown in Fig. 18) with only two targets is considered. Since a target is deactivated once reached, at each time step, only one target is active and the robot is forced to 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. 19, demonstrates the efficiency of learned visual feedback. It can be observed that after learning the robot is able 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. 19 before learning (dashed

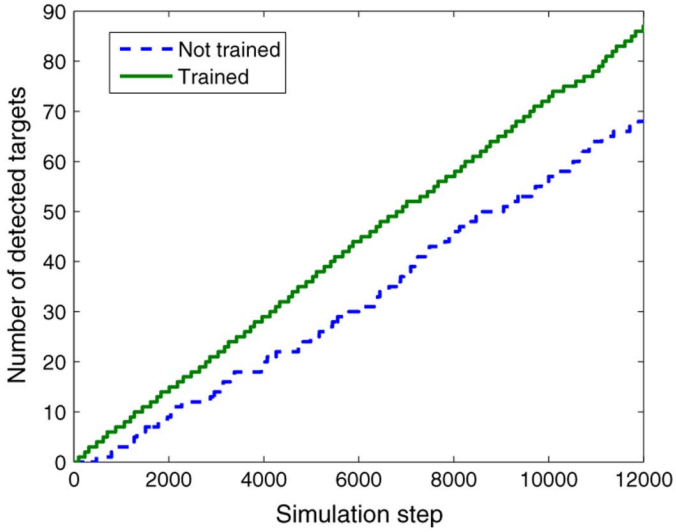


Fig. 16. Number of targets found with or without visual input (environment of Fig. 14).

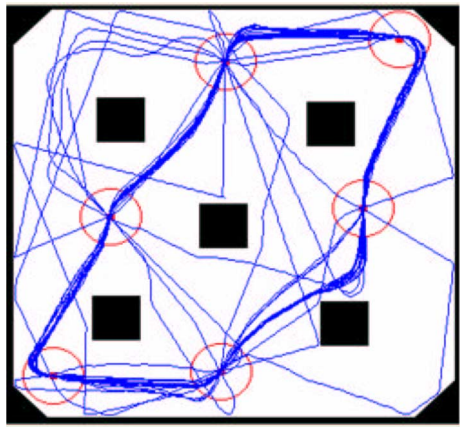


Fig. 17. Long run simulation of target approaching showing how at the end of the simulation the robot follows a stereotyped trajectory, which allows it to visit all the targets avoiding the obstacles.

curve) the robot for a long time interval ($500 < t < 10\,000$ simulation steps) does not find the target hidden by the two walls.

Finally, in Fig. 20, we compare the trajectories obtained before and after learning in the case of a single target. The different trajectories are obtained for different initial positions at the same distance from the obstacle (indicated by the large circle in Fig. 20). As it can be observed, before learning [Fig. 20(a)] the robot is able to find the target only when its trajectory intersects the visibility radius of the target (the small circle in the center of Fig. 20). After learning, the robot finds the target in all cases except one. The trajectories of Fig. 20(b) show that the robot starts turning towards a target at a distance, which is a function of its initial position.

VI. 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

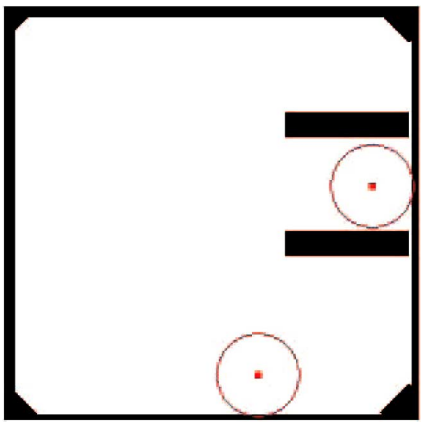


Fig. 18. Environment referred to the second set of simulations for target approaching.

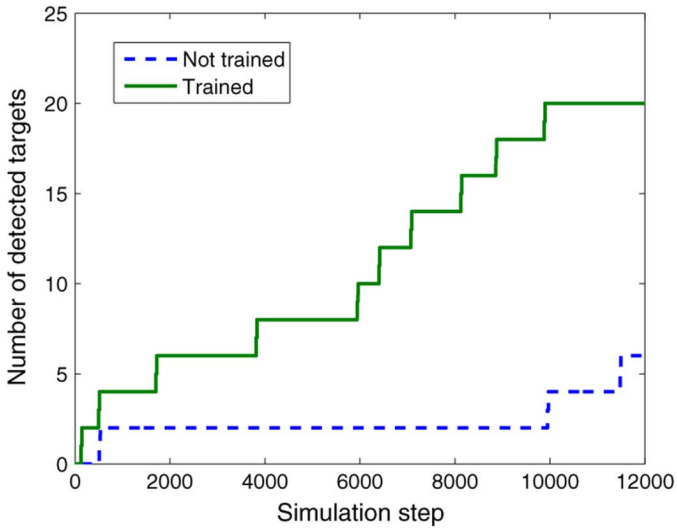


Fig. 19. Number of targets found with or without visual input (environment of Fig. 18).

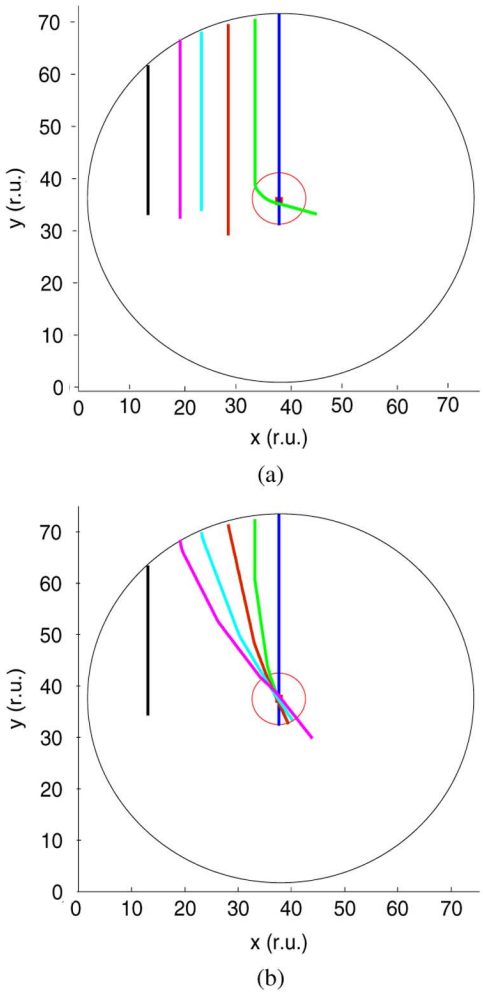


Fig. 20. Trajectories obtained for different initial positions at the same distance from the obstacle: (a) before learning target approaching; (b) after learning.

rats have to find a food reward in a T-maze. In such experiments [1], depending upon training conditions and on "what is invariant throughout the trials" [35], different cognitive responses can be observed: if the food is always in the right arm of the maze, the animal learns 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 is in one of the arms of the T-maze, and a mark object is positioned on the front wall in slightly different positions. Depending on the position of the mark and of the target, the mark object can be considered a meaningful or useless visual cue.

The scheme of the spiking network for this navigation task is shown in Fig. 21. There are now two layers of class I excitable neurons that process the visual input. In particular, the first layer (bottom, left-hand side in Fig. 21) exactly acts as in the network of Fig. 13: neuron spikes code the presence of a red object in the receptive field of the neuron. We assume that it is *a priori* known which objects 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 mark 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 interneurons are updated according to the STDP synaptic rule. By updating these synaptic weights, the robot now has to learn the appropriate turning direction in correspondence of a given mark object. A synaptic weight is increased when after a turn the robot sees the target.

We focus on the different training environments shown in Fig. 22, in order to learn: visual cue-based response [Fig. 22(a)] or egocentric motor response [Fig. 22(b)]. In the last example [Fig. 22(c)], the training conditions are such that neither egocentric motor responses nor cue-based responses can be learned, since target position depends neither on the T-maze arm nor on the position of the visual cue.

In correspondence of the T-bifurcation, there are marks placed in the front wall. These objects are landmark candidates,

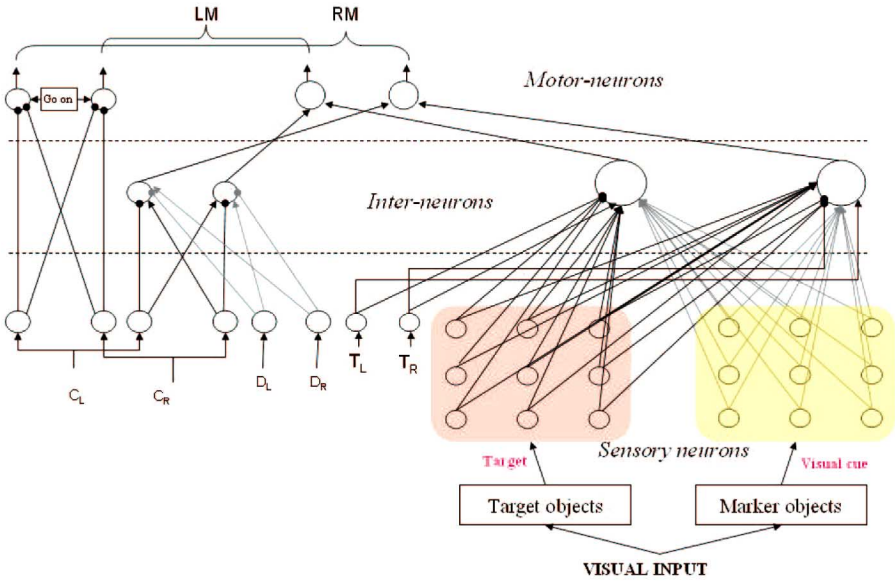


Fig. 21. Scheme of the spiking network for navigation with visual cues.

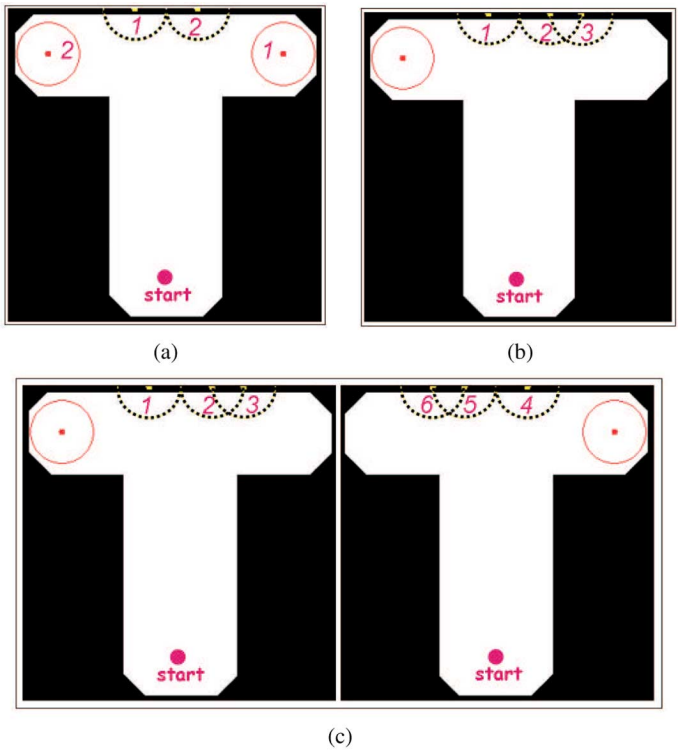


Fig. 22. Training environments to learn associations between visual cue and target. (a) The target is in the opposite with respect to the visual cue. (b) The target is ever at left, independently of the position of the visual cue. (c) The target is not correlated with the position of the dashed circle mark in the wall and may be either at left or right.

since they can became landmarks if, after successive presentations of the environment configuration, the robot discovers a relationship between the mark and the target position. The position of the mark is indicated with a dotted circle in Fig. 22. Several trials with slightly different mark and target positions

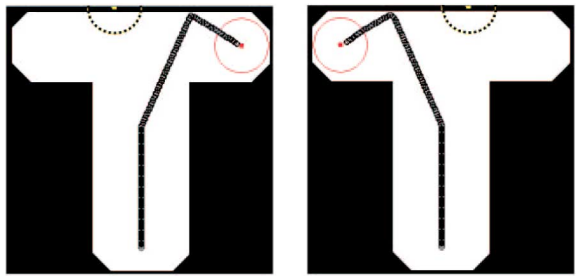


Fig. 23. Trajectories followed by the robot for two different cases. The robot training has been carried out in the environment of Fig. 22(a).

for each training environment are carried out: they are numbered and presented to the robot several times. For example, for the T-maze shown in Fig. 22(a), two trials are carried out: as it can be noticed, if the visual cue is on the right, the target is in the left arm and vice versa. Therefore, in both cases, the target, indicated with a small rectangle in the center of a circle (continuous line), is in the opposite arm with respect to position of the visual cue. After training, the robot learns this association as it can be noticed in Fig. 23: when the mark (dotted curve) is on the left, the robot turns on the right and when the mark is on the right, the robot correctly turns on the opposite side.

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 used in the case of Fig. 22(a) consists of several training cycles, where each training cycle in turn consists of 15 trials with the environment configuration labeled as 1 in Fig. 22(a) and 15 trials with the configuration 2. 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.

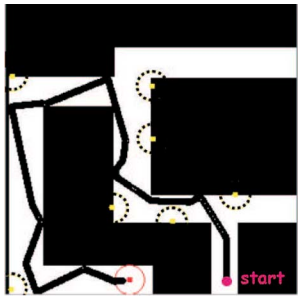


Fig. 24. Trajectory followed by the robot in a maze, which can be solved relying on visual cues. Visual cues are oppositely placed with respect to the right choice as in the training environment of Fig. 22(a).

During the first ten training cycles, the robot 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. 23).

In the case of Fig. 22(b), three different trials are repetitively presented to the robot and the learning protocol is adjusted to match the different number of possible configurations (each training cycle consists of ten trials with configuration 1, ten trials with configuration 2, and ten trials with configuration 3). At the end of the training (four training cycles), the robot learns an egocentric left response. Finally, in the last case [Fig. 22(c)], the robot does not learn any kind of response, since the target is not correlated with the position of the mark in the wall and may be either at left or right. In this case, therefore, the mark object does not become a landmark for the robot, and will be simply ignored by the robot.

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 on the opposite side with respect to the correct arm. We let a simulated robot navigate in this maze; the simulated robot has been trained in the simulation environment of Fig. 22(a), which takes into account the same rule needed to solve the maze. As shown in Fig. 24, the robot is able to successfully navigate in the maze and find the target.

A. *Remarks on Navigation With Visual Cues*

Unlike the case of learning obstacle avoidance, it should be pointed out that in this case, the unconditioned stimulus occurs some time after the turn. For this reason, the parameters of the synaptic rule (5) are chosen as follows: $A_+ = A_- = 0.005$; $\tau_+ = 3000$ ms; $\tau_- = 1000$ ms. Let us consider the plastic synapses between sensory neurons connected to the visual cue and the interneurons. Since the interneurons are activated (i.e., they emit spikes) when the target lies within their receptive field and the sensory neurons are activated when the visual cue falls within their receptive field, Δt in (5) depends on the distance between target and cue. If, for instance, Δt is large (in absolute value) compared to τ_+, Δw is very small and the weights are practically unchanged. Thus, τ_+ and τ_- should be chosen taking into account that the difference between the time when the robot sees the visual cue and the time when the robot finds the target can be large.

The choice of A_+ and A_- influences the learning time, but we observed the same qualitative behavior in a wide range of these parameters. Thus, the choice of these parameters in the three navigation tasks can be easily accomplished.

We remark that the first two examples discussed in this paper are clearly examples of classical conditioning, and the learning strategy based on correlation is unsupervised. In the third example, the robot explores the arena looking for targets and associates cue position with an action, which successfully drives the robot towards the target. In view of the implementation of an autonomous robot, we believe that the important aspect of this algorithm is that in any case the "critic" [36] is internal and thus the algorithm can be considered unsupervised. As such it benefits from the advantages of unsupervised learning algorithms (self-organization, integration of multiple sensors, and adaptation to unforeseen situations).

Another remark concerns the biological plausibility of large time delays in the STDP rule. STDP is a well-demonstrated mechanism accounting for neuronal plasticity, which acts at typical time scales of milliseconds. On the other hand, behavioral sequences occur over arbitrary time scales, for example, classical conditioning often operates with time intervals between CS and US, which are at the time scale of seconds. How to bridge the gap between the time scales of STDP and those of behavioral learning is still a theoretical open problem referred as the time-gap problem [36]. Several potential solutions are reviewed in [37]. One solution is to code either fast or slow sequences at a millisecond resolution, but this has the problem that slow sequences require expensive representations. Another solution is to encode slow sequences in neurons that are part of slow loops firing with slow oscillation frequencies. Another very important mechanism used to explain how the nervous system may learn behavioral sequences of arbitrary time scales with a fixed millisecond rule is introduced by Mehta and colleagues [38]. They identify in the mechanism of phase coding in the hippocampus a key factor to learn sequences of locations with learning rules on the millisecond scale. Mehta *et al.* hypothesize that hippocampal place cells generate spike time codes for position through the interaction of slower rate codes with an oscillatory inhibition.

There are many examples in literature that take into account models reproducing the main features of hippocampal cells to implement navigation control [5]–[9]. These models, often based on experimental evidence in rats [8], [9] identify a key role of the hippocampus for goal-directed spatial navigation: hippocampus acts as an associative memory, where place cells fire at specific spatial locations. Also in human studies, it is demonstrated that short term memories are stored by patterns of neuronal activity [39].

Furthermore, we mention that from another viewpoint, the time-gap problem is solved thanks to the fact that the neural dynamics is also characterized by other time scales than those of milliseconds [40]. This solution is exploited through the paradigm of liquid state machines for the control of a two-jointed arm [41], of a roving robot [42], and for movement prediction [43].

Recently, a solution to the time-gap problem was proposed by Izhikevich [25] based on dopamine-modulated STDP. This solution introduces a slow variable with characteristic time constants of the order of seconds.

The general conclusion that can be derived from all these models is that in any case a slow mechanism should be used and that a realistic model of goal-directed navigation would require the inclusion of other structures like hippocampal cells [9], which are beyond the purpose of our model. Our simplified network was not proposed as a model of the hippocampal function or structure, but was designed starting from simple reactive controllers (such as those discussed in [29]) and extending their function to implement more complex behaviors, keeping in mind the suitability of an easy robot implementation.

VII. Conclusion

In this paper, 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. The simulation results reported in this paper show the efficiency of the algorithm.

The use of spiking neurons to build biologically inspired navigation systems poses new problems and offers several innovative aspects: it adds accuracy to the bioinspired model; it requires a new (biological) paradigm for learning (the STDP); it is based on dynamical systems instead of a static relationship between input and output.

Concerning the level of biological inspiration, we point out that the underlying mechanism of the approach is biologically inspired, but the network (i.e., the control system) is artificial and *ad hoc* designed for a roving robot. In fact, the aim of this paper is not to replicate an existing network of neurons, but to design a suitable controller for robot navigation by using neurons as building blocks for our control system. In this sense, 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.

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 that can be interesting for studies on navigation strategies. The approach is constructive 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 that can be included, for instance, to take into account other visual cues.

Acknowledgment

The authors would like to thank Dr. B. Webb for her collaboration and the peer reviewers for their comments.

References

[1] O. Trullier, S. I. Wiener, A. Berthoz, and J.-A. Meyer, "Biologically-based artificial navigation systems: Review and prospects," *Progr. Neurobiol.*, vol. 51, pp. 483–544, 1997.

[2] R. C. Arkin, *Behaviour Based Robotics*. Cambridge, MA: MIT Press, 1997.

[3] S. Grossberg, W. Maass, and H. Markram, "Introduction: Spiking neurons in neuroscience and technology," *Neural Netw.*, vol. 14, Special Issue on Spiking Neurons, no. 6–7, p. 587, Jul. 2001.

[4] M. O. Franz and H. A. Mallot, "Biomimetic robot navigation," *Robot. Autonom. Syst.*, vol. 30, pp. 133–153, 2000.

[5] N. Burgess, M. Recce, and J. O'Keefe, "A model of hippocampal function," *Neural Netw.*, no. 7, pp. 1065–1081, 1994.

[6] N. Burgess and J. O'Keefe, "Neuronal computations underlying the firing of place cells and their role in navigation," *Hippocampus*, no. 6, pp. 749–762, 1996.

[7] N. Burgess, S. Becker, J. A. King, and J. O'Keefe, "Memory for events and their spatial context: Models and experiments," *Philos. Trans. Roy. Soc. London B*, vol. 356, pp. 1–11, 2001.

[8] R. A. Koene, A. Gorchetchnikov, R. C. Cannon, and M. E. Hasselmo, "Modeling goal-directed spatial navigation in the rat based on physiological data from the hippocampal formation," *Neural Netw.*, vol. 16, pp. 577–584, 2003.

[9] O. Jensen and J. E. Lisman, "Hippocampal sequence-encoding driven by a cortical multi-item working memory buffer," *Trends Neurosci.*, vol. 28, no. 2, 2005.

[10] D. Floreano and C. Mattiussi, "Evolution of spiking neural controllers for autonomous vision-based robots," in *Evolutionary Robotics IV*. Berlin, Germany: Springer-Verlag, 2001.

[11] D. Floreano, Y. Epars, J. C. Zufferey, and C. Mattiussi, "Evolution of spiking neural circuits in autonomous mobile robots," *Int. J. Intell. Syst.*, vol. 21, no. 9, pp. 1005–1024, 2006.

[12] P. F. M. J. Verschure, B. J. A. Kröse, and R. Pfeifer, "Distributed adaptive control: The self-organization of structured behavior," *Robot. Autonom. Syst.*, vol. 9, pp. 181–196, 1992.

[13] P. F. M. J. Verschure, T. Voegtlin, and R. J. Douglas, "Environmentally mediated synergy between perception and behaviour in mobile robots," *Nature*, vol. 425, pp. 620–624, 2003.

[14] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent plasticity," *Nature Neurosci.*, vol. 3, pp. 919–926, 2000.

[15] E. M. Izhikevich, J. A. Gally, and G. M. Edelman, "Spike-timing dynamics of neuronal groups," *Cerebral Cortex*, vol. 14, pp. 933–944, 2004.

[16] I. P. Pavlov, *Conditioned Reflexes*. London, U.K.: Oxford Univ. Press, 1927.

[17] G. M. Shepherd, *Neurobiology*. London, U.K.: Oxford Univ. Press, 1994.

[18] S. Song and L. F. Abbott, "Cortical development and remapping through spike timing-dependent plasticity," *Neuron*, vol. 32, pp. 339–350, 2001.

[19] C. Leibold, R. Kempter, and J. L. van Hemmen, "Temporal map formation in the barn owl's brain," *Phys. Rev. Lett.*, vol. 87, no. 24, pp. 248101-1–248101-4, 2001.

[20] W. Senn and N. J. Buchs, "Spike-based synaptic plasticity and the emergence of direction selective simple cells: mathematical analysis," *J. Comput. Neurosci.*, vol. 14, pp. 119–138, 2003.

[21] C. Leibold and J. L. van Hemmen, "Temporal receptive fields, spikes, and Hebbian delay selection," *Neural Netw.*, vol. 14, pp. 805–813, 2001.

[22] S. Fusi, "Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates," *Biol. Cybern.*, vol. 87, pp. 459–470, 2002.

[23] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[24] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[25] E. M. Izhikevich, "Solving the distal reward problem through linkage of STDP and dopamine signaling," *Cerebral Cortex Advance*, vol. 17, pp. 2443–2452, 2007.
[26] P. F. M. J. Verschure and R. Pfeifer, "Categorization, representations, and the dynamics of system-environment interaction: A case study in autonomous systems," in *From Animals to Animats: Proc. 2nd Int. Conf. Simul. Adapt. Behav.*, J. A. Meyer, H. Roitblat, and S. Wilson, Eds., 1992, pp. 210–217.
[27] P. Arena, L. Fortuna, M. Frasca, G. Lo Turco, L. Patané, and R. Russo, "A new simulation tool for action-oriented perception systems," in *Proc. 10th Int. Conf. Emerging Technol. Factory Autom.*, Catania, Italy, Sep. 19–22, 2005, pp. 571–577.
[28] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.
[29] B. Webb and T. Scutt, "A simple latency dependent spiking neuron model of cricket phonotaxis," *Biol. Cybern.*, vol. 82, no. 3, pp. 247–269, 2000.
[30] K. P. Körding and P. Konig, "A spike based learning rule for generation of invariant representations," *J. Physiol. (Paris)*, vol. 94, no. 5–6, pp. 539–548, 2000.
[31] K. P. Körding and P. Konig, "A learning rule for dynamic recruitment and decorrelation," *Neural Netw.*, vol. 13, no. 1, pp. 1–9, 2000.
[32] M. A. Sanchez-Montanes, P. F. M. J. Verschure, and P. Konig, "Local and global gating of synaptic plasticity," *Neural Comput.*, vol. 12, no. 3, pp. 519–529, 2000.
[33] Robot movies web page, [Online]. Available: http://www.spark.diees.unict.it/STDP.html
[34] P. Arena, A. Basile, M. Bucolo, and L. Fortuna, "An object oriented segmentation on analog CNN chip," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 7, pp. 837–846, Jul. 2003.
[35] F. Restle, "Discrimination of cues in mazes: A resolution of the 'place-vs-response' question," *Psychol. Rev.*, vol. 64, no. 4, pp. 217–228, 1957.
[36] F. Worgotter and B. Porr, "Temporal sequence learning, prediction and control—A review of different models and their relation to biological mechanisms," *Neural Comput.*, vol. 17, pp. 245–319, 2005.
[37] O. Melamed, W. Gerstner, W. Maass, M. Tsodyks, and H. Markram, "Coding and learning of behavioral sequences," *Trends Neurosci.*, vol. 27, no. 1, pp. 11–14, 2004.
[38] M. R. Mehta, A. K. Lee, and M. A. Wilson, "Role of experience and oscillations in transforming a rate code into a temporal code," *Nature*, vol. 417, pp. 741–746, 2002.
[39] J. E. Lisman and M. A. Idiart, "Storage of 7 ± 2 short-term memories in oscillatory subcycles," *Science*, vol. 267, no. 5203, pp. 1512–1515, 1995.
[40] W. Maass, T. Natschlager, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, pp. 2531–2560, 2002.
[41] P. Joshi and W. Maass, "Movement generation with circuits of spiking neurons," *Neural Comput.*, vol. 17, no. 8, pp. 1715–1738, 2005.
[42] H. Burgsteiner, "Imitation learning with spiking neural networks and real-world devices," *Eng. Appl. Artif. Intell.*, vol. 19, no. 7, pp. 741–752, 2006.
[43] H. Burgsteiner, M. Kroell, A. Leopold, and G. Steinbauer, "Movement prediction from real-world images using a liquid state machine," *Int. J. Appl. Intell.*, vol. 26, pp. 99–109, 2006.
[44] OpenGL Library [Online]. Available: http://www.opengl.org/

Paolo Arena (S'93–M'97–SM'01) received the degree in electronic engineering and the Ph.D. degree in electrical engineering from the University of Catania, Catania, Italy, in 1990 and in 1994, respectively.

Currently, he is an Associate Professor of System Theory and Biorobotics at the University of Catania. He published more than 220 technical papers, six books, and several international patents. His research interests include adaptive and learning systems, nonlinear systems, spiking and artificial neural networks, cellular neural networks, collective behavior in living and artificial neural structures, and cognitive systems.

Dr. Arena served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS.

Luigi Fortuna (M'90–SM'99–F'00) was born in Siracuse, Italy, in 1953. He received the degree of electrical engineering (*cum laude*) from the University of Catania, Catania, Italy, in 1977.

Currently, he is Full Professor of System Theory at the University of Catania. He now teaches the following courses: system theory, adaptive systems, and biomedical engineering. He was the coordinator of the courses in electronic engineering and Head of the DIEES Department. Since 2005, he has been the Dean of the Engineering Faculty of the University of Catania. He has published more than 450 technical papers and is coauthor of nine scientific books: *Bio-Inspired Emergent Control of Locomotion Systems* (Singapore: World Scientific, 2004), *Soft- Computing* (New York: Springer-Verlag, 2001), *Nonlinear Non Integer Order Circuits and Systems* (Singapore: World Scientific, 2001), *Cellular Neural Networks* (New York: Springer-Verlag, 1999), *Neural Networks in Multidimensional Domains* (New York: Springer-Verlag, 1998), *Model Order Reduction in Electrical Engineering* (New York: Springer-Verlag, 1994), *Robust Control—An Introduction* (New York: Springer-Verlag, 1993). His scientific interests include: robust control, nonlinear science and complexity, chaos, cellular neural networks, soft-computing strategies for control, robotics, micro/nanosensor and smart devices for control, and nanocellular neural networks modeling.

Dr. Fortuna is the IEEE Fellow. He was the IEEE Circuits and Systems Society (CAS) Chairman of the Cellular Neural Network (CNN) Technical Committee, the IEEE CAS Distinguished Lecturer 2001–2002, and the IEEE Chairman of the IEEE CAS Chapter Central-South ITALY.

Mattia Frasca (S'01–M'03) was born in Siracusa, Italy, in 1976. He graduated in electronics engineering in 2000 and received the Ph.D. degree in electronics and automation engineering in 2003 from the University of Catania, Catania, Italy.

Currently, he is a Research Associate at the University of Catania. He teaches the course "Optimization techniques for control" at the Faculty of Engineering. He is involved in many research projects and collaborations with industries and academic centres. He is coauthor of two research monographs (World Scientific): one on locomotion control of bioinspired robots and one on self-organizing systems. He published more than 100 papers in refereed international journals and international conference proceedings and is coauthor of two international patents. His scientific interests include nonlinear systems and chaos, cellular neural networks, complex systems, and bioinspired robotics.

Dr. Frasca is a referee for many international journals and conferences. He was in the organizing committee of the Tenth "Experimental Chaos Conference."

Luca Patané (S'01) was born in Catania, Italy, in 1978. He graduated in computer science engineering in 2001 and received the Ph.D. degree in electronics and automation engineering in 2005 from the University of Catania, Catania, Italy.

Currently, he is a Research Associate at the University of Catania. His scientific interests include cellular neural networks, complex systems, bioinspired robotics, and perception. Recently, his research activity has been focused on the formalization of an insect brain computational model.