Perception for action: dynamic spatiotemporal patterns applied on a roving robot

Paolo Arena , Luigi Fortuna, Davide Lombardo and Luca Patanè

Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi, Università degli Studi di Catania, viale A. Doria 6, 95125 Catania, Italy. E-mail: [parena, dlombardo]@diees.unict.it , Tel: +39-095-7382322, Fax: +39-095-330793.

Abstract

In this paper we present the application of a bio-inspired control architecture to a roving robot performing different tasks. The key of the control system is the perceptual core: here heterogeneous information coming from sensors is merged to build an internal portrait representing the current situation of the environment. The internal representation triggers an action as the response to the current stimuli closing the loop between the agent and the external world. The robot internal state is implemented through a nonlinear lattice of neuron cells allowing the generation a large amount of emergent steady state solutions under the form of Turing Patterns, which are incrementally shaped, through learning, so as to constitute a ”mirror” of the environment conditions. Reaction-Diffusion Cellular Nonlinear Networks were chosen to generate Turing patterns as internal representations of the robot surroundings. The associations between incoming sensations and the perceptual core and between Turing patterns and actions to be performed, are driven by two reward-based learning mechanisms. Simulation results and experiments on a roving robot are reported to show the suitability of the approach.

Index Terms

Nonlinear dynamical systems, reward-based learning, perception, autonomous navigation, Turing patterns.

I. INTRODUCTION

Patterns and schemes are ubiquitous in Nature: from seashells and animal coats, to gastrulation, patterns play a fundamental role in life. For example, animals can easily and promptly recognize preys or predators from a smell or visual and acoustic information. All these sensations are nothing else than spatial temporal patterns eliciting instinctive or learned actions, which guarantee the animal life (Kelso, 1995). Our methodology is inspired by the idea that the sensing-perception-action cycle is mediated through spatial-temporal patterns: actions are planned and executed following a pattern flow, which is continuously created and modified via the feedback through the environment.

We hypothesize that spatial-temporal patterns, under the form of Turing patterns (Turing, 1952), take place also in the internal neural structure devoted to plan actions.

This idea is partially supported by experiments in simple animals, for example in the *clione Limacina* (Orlovsky, Deliagina & Grillner, 1999), where sensory-induced motion or behavior variations are caused by a modulation in neural patterns of activity in non-spiking neuron populations. Under these considerations, perception is here considered, implemented and realized through a pattern forming process, driven by the environment and modulated by an unsupervised learning mechanism. From the formal point of view, one direct way to model such dynamics comes from Alan Turing's theory (Turing, 1952)of morphogenesis. On the wave of this theory, morphogenetic pattern formation was subsequently mathematically formulated through non linear PDEs, the so called reaction-diffusion equations (Murray, 2002). These were represented via Cellular Neural Networks (CNN), leading to the Reaction-Diffusion CNNs (RD-CNNs)(Chua & al, 1995). Moreover, such structures were successfully implemented through programmable analog-logic microprocessors, able to solve large scale non linear PDEs in real time (Arena & Fortuna, 2002, Adamatzky et al., 2004). This led to the use of RD-CNNs to generate Turing

Patterns (Manganaro, Arena & Fortuna, 1999). As previously said, an RD-CNN represents a non linear PDE where patterns emerge as steady state solutions. The idea behind the use of Turing patterns for perception lies in exploiting pattern emergence by adding simple learning to modulate the geometry of the basins of attraction of the patterns. As a consequence, after learning, each pattern represents an environmental situation, and this pattern elicits a specific action which maximizes a pre-defined Reward Function. This function returns a value proportional to the degree of satisfaction related to the robot mission.

This methodology was introduced in order to model the sensing-perception-action cycle, so the potential applications are manifold. In this paper we introduce the first and most direct application to the learning of navigation control mechanisms for a roving robot, both in simulation and in real world experiments.

For the autonomous navigation of a robot, the ability to correctly interpret incoming stimuli is relevant since the presence of unpredictable situations and dynamically changing environments makes it impossible to pre-design the robot behavior only on the basis of *a priori* knowledge. To meet these needs, traditional AI tends to build a consistent and complete representation of real world. This requires an huge amount of resources and does not take into careful consideration those perceptual states which specifically depend on the behavioral needs of a robot, i.e. on the specific task to be performed.

To overcome these problems, in the last years the machine perception researchers developed a new paradigm which considers perception as a process tightly connected to the motor actions (Arkin, 1997). Perception is now considered as a process indivisible from action: behavioral needs provide the context for the perceptual process, which, in turn, works out the information required for motion control (Arkin, 1997). In this view, internal representations are compact and abstract models built on the basis of what is really needed for the agent to achieve its behavioral tasks (Brooks, 1994) and this process is mediated through a behavioral-dependent internal state (Nolfi, 2002).

Following this new prospective, we refer here to *representation* as the internal state which results from the dynamical processing of the sensory events which lead to a Turing pattern. This shows an abstract picture of the characteristics of the environment in which the robot is situated.

Aiming to solve the problem of autonomous robot navigation, without *a priori* knowledge, the successful interaction between a robot and its surroundings is built through skill-based (Tani & N. Fukumura, 1994a, Tani & N. Fukumura, 1994b) learning mechanisms. These allow the robot to achieve its tasks by building both an adequate association between sensory events and internal representation and a suitable state-action map.

In this paper we describe how our bio-inspired control architecture, whose theoretical background has been deeply explained in (Arena et al., 2007a), can be applied to control a roving robot both in simulation and in real (world) experiments, to deal with some of the traditional tasks of bio-inspired navigation (Franz & Mallot, 2000) such as direction-following and target aiming while avoiding obstacles.

The control architecture is divided into functional blocks. The first is the *Sensing Block*, where external stimuli are plastically divided in classes which are used as initial conditions for a two-layer RD-CNN (Manganaro, Arena & Fortuna, 1999): this is the core of the perceptual process. The CNN parameters are chosen appropriately for generating Turing patterns (Goras & Chua, 1995) and (Arena et al., 2007a), which are considered as an abstract representation of the surrounding environment. Each pattern is associated with an action by an *Action Selection Network*. To accomplish its task, the robot is provided with no *a priori* knowledge on how to perform it and learns by trial and error, like in the paradigm of operant conditioning (Skinner, 1974). Learning is implemented by two mechanisms: an unsupervised learning acts at the *Sensing Block,* allowing the system to modulate the basins of attraction of the Turing patterns, while a simple reward-based reinforcement learning builds associations between Turing patterns and actions.

Instead of using an artificial neural network (Verschure, Voegtlin & Douglas, 2003), our idea is to introduce a nonlinear dynamical system as the perceptual core of the whole architecture. This is due not only to biological plausibility, but also to the high level of plasticity which can be reached thanks to the modulation of the basins of attraction and to the high number of patterns which can be generated by the RD-CNN (Arena et al., 2007a). Furthermore, we do not strictly follow the paradigm of classical conditioning (Pavlov, 1927): we let the actions vary as the result of a simple, but effective learning mechanism, which improves the versatility of the methodology.

In the next section we will describe the application of the control architecture to a roving robot, in section III we will present some simulation results, while experiments on a roving robot are shown in section IV. Finally, in section V we will draw the conclusions and present the further developments which we are investigating.

II. ROBOT CONTROL ARCHITECTURE

Figure 1 about here.

The robot control architecture is a simplified version of the architecture proposed in a previous work (Arena et al., 2007a) and is made-up of four main blocks (Fig. 1): the *Sensing Block*, where patterns arise to represent environmental situations; the *Perception Block*, which forms a representation of the environment on the basis of the processed stimuli; the *Action Selection Network*, which triggers actions to the effectors; the *Difference of Reward Function (*DRF*) Block*, which evaluates the “goodness” of an action and drives the learning process. In the following they are described in more details.

A. Sensing Block

To face with the navigation learning, the robot is provided with three distance sensors (covering the front, the left and the right side of the robot respectively) for the obstacle detection (Fig. 2a). In the real robot, the front distance sensor, due to its crucial role, is implemented by mediating the information coming from two sensors as explained in the following. Furthermore, the rover receives information on the angle between the robot orientation and the direction robot-target. This information could come in different ways, depending on the specific task: it could come from a network inspired by cricket phonotaxis (Webb & Scutt, 2000) which processes the auditory stimuli, or it could be extracted from the visual field by using landmarks (Arena et al., 2007b).

The sensorial inputs are all scaled in the range $[-1, 1]$ by a linear function. Each sensing stimulus is the input for a Sensing Neuron (SN) with piece-wise linear activation function made-up of amplitude-varying steps (Fig. 3). The step amplitudes are learned in an unsupervised way, as it will be outlined in a following subsection.

Finally, each output of the SNs sets the initial conditions for a cell of the RD-CNN representing the perceptual core.

B. Perceptual core

Figure 2 about here.

The whole perceptual process has the aim to transform different environmental situations into representations, triggering specific actions. These mechanisms are plastically modified by experience.

Obviously, the creation of a concise representation of the environment is crucial for the perceptual process, since this is the result of the dynamic processing of external stimuli. To implement this feature, we use a nonlinear dynamical system, a two-layer 4×4 RD-CNN, able to generate Turing patterns (Turing, 1952). The dimension of the network has been fixed to 4×4 on the basis of our previous work (Arena et al., 2007a). Each cell (i, j) of the two-layer RD-CNN has two state variables ($x_{1;i,j}$ for the first layer and $x_{2;i,j}$ for the second layer with $i, j = 1, .., 4$) and reads:

$$\begin{aligned} \dot{x}_{1;i,j} &= -x_{1;i,j} + (1+\mu+\varepsilon)y_{1;i,j} - sy_{2;i,j} + D_1\nabla^2 x_{1;i,j} \\ \dot{x}_{2;i,j} &= -x_{2;i,j} + sy_{1;i,j} + (1+\mu-\varepsilon)y_{2;i,j} + D_2\nabla^2 x_{2;i,j} \\ y_{h;i,j} &= \tfrac{1}{2}(|x_{h;i,j}+1| - |x_{h;i,j}-1|) \qquad\qquad h = 1,2 \end{aligned} \tag{1}$$

where $y_{h;i,j}$ ($h = 1, 2$) is the output of the layer h of the cell (i, j) and D_1, D_2, μ, ε and s are parameters of the model.

To obtain the emergence of Turing patterns, it is necessary that some of the spatial modes, related to the chosen geometry, are within a "Band of unstable modes" (Bu), i.e. correspond to positive temporal eigenvalues. The conditions to obtain Turing patterns are (Murray, 2002):

$$\begin{cases} \mu < 0 \\ \mu^2 + s^2 > \varepsilon^2 \\ \varepsilon > -\mu\frac{d+1}{d-1} \\ \frac{[d(\mu+\varepsilon)+(\mu-\varepsilon)]^2}{4d} > \mu^2 - \varepsilon^2 + s^2 \end{cases} \tag{2}$$

where $d = \frac{D_2}{D_1}$. To satisfy the above discussed conditions, the chosen parameters have been $\mu = -0.7$, $\varepsilon = 1.1$, $s = 0.9$, $D_1 = 0.05$, $D_2 = 15$.

As shown in Fig. 2.b, the output of each SN sets the initial conditions for the state variable of the first layer for one of the corner cells, which have been proven to have higher control on the pattern generation than the other cells. In fact, the corner cells, by imposing *zero-flux* boundary conditions, are influenced only by two neighbor-cells. Anyway, the architecture can be expanded to other sensors by setting the initial conditions of one or two of the other cells. The initial conditions for the state variable of the first layer for all the other cells are set to zero. The initial conditions for the state variable of the second layer are randomly chosen in the range $[-0.005, 0.005]$ for all the cells.

The RD-CNN evolves towards the condition in which all the state variables of the first layer, i.e. the

$x_{1;i,j}$, saturate at a value above 1 or below -1. In this case, each output variable $y_{1;i,j}$ will be either 1 or -1, a condition that we consider a Turing pattern. In the simulations, we choose an integration constant $dt = 0.001$ and a number of integration steps $N = 100000$, enough to reach the above described steady state condition.

To simplify the following processing, we associate a simple integer code for each Turing pattern:

1) the first-layer cells are enumerated starting from the upper left-hand corner.
2) a symbolic value $y_{simb,c}$ is associated with each cell c as follows:
   - if the cell output is $y_{1,c} = -1$ then $y_{simb,c} = 0$
   - if the cell output is $y_{1,c} = 1$ then $y_{simb,c} = 1$
3) an integer code is associated with the steady-state pattern:

$$code = \sum_{c=0}^{15} y_{simb,c} 2^c \tag{3}$$

It is to be noticed that Turing patterns, by definition, lead to output cells whose value are only ± 1. To finely represent different environmental situations, it would be feasible to have a large number of Turing patterns, but the pattern control would risk to be less effective. A trade-off between the amount of different patterns and easiness of control has to be found. The number of the emerging Turing patterns can be modified tuning the parameter $\gamma = \frac{1}{D_1}$: by increasing such parameter, Bu becomes wider, increasing the number of possible modes to be selected (Arena et al., 2007a). In this way, a stronger competition between the allowed modes is triggered to generate patterns. In this paper, setting $\gamma = 20$, the dispersion curve hosts 12 unstable modes.

Once processed the external stimuli, we reset the CNN, set the initial conditions for the corner cells through the outputs of the SNs and let the CNN evolve and generate a Turing pattern. Its code is stored in a *Pattern Vector* at the first occurrence. Each element of the pattern vector contains the *Pattern Code* and the step of

its last occurrence (*Occurrence Lag*). The Pattern Vector can contain up to 2000 items: this number could seem to be very large, but in the first part of the learning phase, the number of emerged patterns is very high, so the Pattern Vector should be large enough to avoid that the frequent substitution of patterns in the table would cause the loss of the learning information. If the pattern vector is full, the new element replaces the least recently used (LRU), i.e. that one with the lowest *Occurrence Lag* value.
The use of the steady states of a dynamical system implies a form of sensor fusion, i.e. we synthesize lots of heterogeneous sensor information into a single attractor. At each step, the information coming from sensors is fused to form a unique abstract representation of the environment.

C. The Action Selection Network and the DRF Block

The Action Selection Network associates each element q of the pattern vector with an action A_q. An action consists of two elements, the module and the phase of the robot movement setting, respectively, the translational step and the rotation. Each element q of the pattern vector is connected to two weights, $w_{q,m}$ and $w_{q,p}$, representing, respectively, module and phase of the action A_q. In this paper, we keep fixed the weight $w_{q,m} = w_m$ for all the patterns and vary only the $w_{q,p}$ through a reward-based reinforcement learning implemented by a simplified Motor Map (MM) (Schulten, 2002], [Arena et al., 2004). Emulating the associative learning in animals, we determine the goodness of an action by means of a Reward Function (RF) defined as follows:

$$RF = -\sum_i k_i \cdot f_i(e^{D_i}) - h \cdot f_T(|\phi_T|) \tag{4}$$

where D_i is the distance between the robot and the obstacle detected by the sensor i ($i = Front(F), Right(R), Left(L)$), Φ_T is the angle between the robot orientation and the direction connecting robot and target, while k_i and h are positive constants determined in a design phase. The RF indicates the task assigned to the robot: in this way it knows *what to do* but not *how to do* that. The RF is used to learn the sensing-perception-action cycle by driving directly the association between Turing patterns and actions and indirectly modulating the basins of attraction of the Turing patterns. This modulation is realized by training the SN layer. In fact the RD-CNN is exploited for it capability to rapidly converge to steady state patterns, with associated basins of attraction. These ones typically depend on the CNN parameters, and so are fixed. In order to add plasticity to them, we act on the afferent association between the stimuli and the CNN itself, by means of a simple associative learning aimed to establish the correct association between the sensory events and the internal representations. The scope of the learning algorithm is to maximize the RF: small absolute values in (4) indicate good situations for the robot. The goodness of an action performed at the step t, is provided by $DRF(t) = RF(t) - RF(t-1)$. A positive (negative) value for DRF indicates a successful (unsuccessful) action. Successful actions are followed by reinforcement, like in the Skinner's experiments (Skinner, 1974).

D. Unsupervised learning in the Sensing Block

SNs are responsible for transforming the incoming stimuli into initial conditions for the RD-CNN which will converge to a Turing pattern.

Figure 3 about here.

Our choice for the SNs activation function consists in an increasing function constituted of ten variable amplitude steps, θ_i ($1 \leq i \leq 10$), covering the whole input range $[-1, 1]$ (Fig. 3). At the beginning of the learning phase, all the steps have zero amplitude. At each step, if the performed action has positive effects ($DRF > 0$), then the step amplitude does not change. Otherwise, when the action results to be negative, the step amplitudes are modified randomly (in order to modulate the basins of attraction for the patterns). The idea is that, when the action associated with the previous situation is no longer able to make the robot succeed in accomplishing the current task, a new pattern (situation) should emerge and the suitable action to this new environmental condition has to be learned by the robot. In such a way the sensorial stimuli will be divided into classes, associating different situations with patterns that generate positive actions. The system constituted by the RD-CNN plus the SNs with simple learning can be seen as a unique perceptual system where the pool of neurons within the CNN generates patterns, as steady state solutions of neural lattices. The SNs learn to suitably associate set of environmental conditions to a given pattern.

More in detail, if the action associated with the currently emerged pattern is unsuccessful (i.e. $DRF < 0$), then the learning algorithm for each SN acts as follows:

- determine which of the RF component has suffered the highest decrease (e.g. the component associated with the Front side obstacle detector)
- for the selected SN determine the step amplitude θ_i related to the current input value;
- extract a number rnd from a zero-mean, uniformly distributed random variable r;
- if rnd is positive, the ten step amplitudes θ_j are modified as:

$$\begin{cases} \theta_j(new) = \theta_j(old) & if \quad j < i \\ \theta_j(new) = \theta_j(old) + rnd & if \quad j \geq i \end{cases} \tag{5}$$

- instead, if rnd is negative:

$$\begin{cases} \theta_j(new) = \theta_j(old) - |rnd| & if \quad j \leq i \\ \theta_j(new) = \theta_j(old) & if \quad j > i \end{cases} \tag{6}$$

To guarantee the convergence of the algorithm, the variable r varies in the range [-h,h] where h, initially sets to 0.5, decreases at each step with an aging coefficient $h_{new} = 0.999 \cdot h_{old}$. The result is that the association between sensorial stimuli and Turing patterns is dynamically tuned by modulating the basins of attraction of the steady state patterns in analogy with the most recent studies and related hypotheses in neurobiology (Freeman, 2004), which show how the process of learning situations is obtained modulating the basins of attraction in the dynamical state space of the brain.

More details on the whole mathematical model are given in (Arena et al., 2007a).

III. SIMULATIONS AND RESULTS: SIMULATED ROBOT

The proposed architecture was tested first of all with a virtual agent and then through a roving robot.

A. Simulation environment

The software simulation environment, developed in $C++$, allows to create an arena constituted by walls, obstacles and targets. Moreover, in the arena a robot equipped with a distributed sensory system can be simulated.

The dimensions of the arena are 489×185 pixels and it is filled by obstacles and a target source. The robot is equipped with three distance and one target sensors. The front side sensor detects obstacles within a limited range of 40 pixels and a visual field of $[-45°, 45°]$ with respect to the robot longitudinal axis. The other two obstacle sensors have visual field of $[-45°, 45°]$ with respect to the direction orthogonal to each robot side and a detection range of 20 pixels. It is to be noticed that, for all the distance sensors, the output is saturated to the limit of the detection range, so even if no obstacles are detected, the output of the sensor would be $40 pixels$ for the front distance sensor, and $20 pixels$ for the other two distance sensors. The target sensor has an unlimited range and provides the angle between the robot orientation and the robot-target direction. All the sensor outputs are scaled in the range $[-1, 1]$. The component of the RF (4) are defined as:

- $f_F(e^{D_F}) = -(1/(e^{\gamma_F * (D_F+1)})$
- $f_L(e^{D_L}) = -(1/(e^{\gamma_L * (D_L+1)})$
- $f_R(e^{D_R}) = -(1/(e^{\gamma_R * (D_R+1)})$

- $f_T(\Phi_T) = -1 * |\Phi_T|$

where $\gamma_F = 2$, $\gamma_L = \gamma_R = 3$ and D_F, D_R, D_L are the scaled distances detected by the sensors. In the following simulations, the choice for the other parameters in (4) was: $k_F = 700$, $k_L = k_R = 400$ and $h = 20$. In this way more importance is given to the contribution of the obstacle information than to the target one, because the former is crucial to preserve the robot integrity. In particular the output coming from the front side obstacle sensor has the greatest weight in the RF. Through the definition of this reward function, we give to the robot knowledge about the task to be fulfilled, but it has no *a priori* knowledge about the correct way to interact with the environment. So the phase of the actions associated with each pattern is randomly initialized within the range $[-20°, 20°]$.

B. Learning phase

As far as the simulated robot is concerned, the task given to the robot consists in aiming a target avoiding obstacles. The learning phase is trained for 200 trials, where a trial is made of a limited sequence of actions (500) allowing to reach the target. The arena used during the training phase consists on sparse obstacles randomly distributed and a target source which is randomly placed at the beginning of each trial.

At the beginning of the learning phase, the robot performs random actions due to the random initialization of the phase weights $w_{p,q}$, which determine the robot heading. During the learning process, the Motor Map-like algorithm corrects the action associated with each pattern. After about 4-6 trials, the robot learns to maintain the target heading and the basins of attractions of the patterns used to obtain this behavior are already defined. Avoiding obstacles needs more trials both for defining the actions and for the modulation of the basins of attractions of the corresponding emerging patterns.

It is to be noticed that almost all the learning sessions that were performed lead to the results described below and in particular in all the sessions, after the learning phase, the simulated robot has acquired the ability to reach the target avoiding obstacles. Mainly, two patterns emerged to reach the target in absence of obstacles. In fact, when aiming the target, the approach is done by zig-zag movements by using two patterns which have learned symmetric actions and which emerge alternatively. When the robot is far from heading the target, the same pattern, q, emerges, making the robot execute the same action, (w_m, $w_{q,p}$): this means turning toward the target, until Φ_T decreases under the effect of the performed action. After that, the two patterns begin to alternate generating the zig-zag approaching movement. Another result of the learning is that the robot becomes able to turn towards the target choosing the direction with the minimum angular distance.

The number of patterns associated with the avoidance behavior is greater because we deal with more complex configurations of stimuli. This also depends on the environmental conditions which have been presented during learning. In most cases, the trajectories followed by the robot result to be smooth. Fig. 4 shows a representative result of the learning of the amplitude steps for all the SNs activation functions.

Figure 4 about here.

C. Testing phase

Once having learned the basins of attraction and the actions associated to the emerged patterns, many simulations have been performed to demonstrate the ability acquired by the robot.

To evaluate quantitatively the level of optimization in aiming the target, slightly modifying the original definition (Schul, 1998), we adopted a parameter, called *directedness*. Firstly, we evaluate the distance d_i covered after the execution of a single step and the component of the movement on the cartesian axis:

$$\begin{aligned} d_i &= \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2} \\ \sin_i &= (y_{i+1}-y_i)/d_i \\ \cos_i &= (x_{i+1}-x_i)/d_i \end{aligned} \tag{7}$$

and finally the *directedness* (dir) is defined as follows:

$$dir = \sqrt{[(\sum_{i=1}^{N_S-1} \sin_i)^2 + (\sum_{i=1}^{N_S-1} \cos_i)^2]/(N_S-1)} \tag{8}$$

where N_S is the number of steps performed. The *directedness* varies within the range $[0,1]$: the maximum value is obtained when the robot approaches the target with a straight line.

Figure 5 about here.

The first test demonstrates the easily learned ability to reach the target in absence of obstacle covering a zig-zag trajectory due to the two patterns (codes 12703 and 37156) which drive the behavior (Fig. 5.b). The value of the *directedness*, which in the first learning trial was 0.33 (Fig. 5.a) in this case is $dir = 0.933$ and indicates a good target heading.

Figure 6 about here.

In the second test, the robot had to reach a target placed behind it so it has first to turn and then to aim the target. Fig. 6 shows the trajectory, which in the first part is driven by the pattern with code 12703 whose associated action is turning to left. Once oriented correctly, the robot reaches the target through the zig-zag movement, result of the alternation of the pattern 12703 and 37156. One interesting result of the learning processes is that the two patterns have, as associated phase, around $8.6°$ and $-8°$ respectively, i.e. they are almost symmetric. In fact, it is extremely rare that the robot learns a unique pattern able to let it direct without steering to the target. Instead, a couple of symmetric patterns is feasible, and it is also the most robust solution against noise and disturbance, odometric errors, and so on. It is also the strategies followed by several insect species like crickets (Webb & Scutt, 2000). This, in fact, are well known to adopt a zig-zag movements in performing forward motion.

At the beginning of the test, the phase between the robot orientation and the robot-target direction is $100°$. The robot turns to the side with minimum angular distance driven by the same pattern for many steps allowing a smooth trajectory.

Figure 7 about here.

The third test consists in reaching the target, avoiding a frontal obstacle as shown in Fig. 7. As soon as the obstacle becomes visible by the front side sensor (as seen in Fig. 7.b), at position A the robot front side sensor detects an obstacle at a distance of about $32 pixels$ (Fig. 7.a upper left side). In this condition, the sensor output migrates from the ninth to the eighth step of its correspondent SN activation function, eliciting the emergence of the pattern 63884 the pattern 63884 emerges, triggering a *turn to left* action (about $42 degrees$). The following actions are triggered by two patterns because the obstacle is alternatively visible by the front and right side sensors. When the obstacle is no longer visible, the robot directs toward the target. During learning, it happens that the robot has been very close to an obstacle, so more than one pattern have emerged to deal with different distances from frontal obstacles causing a turning to the left or the right direction. In any case, when the obstacle is at the same time seen by the front and the left (right) side sensors, the actions learned are to turn to right (left).

Figure 8 about here.

It should be noticed that the patterns driving the behavior when the obstacle is seen by the front and the right (left) side sensor, or only by the right (left) side sensor are different, even if they trigger an action of turning to left (right) direction. In fact, the action associated with the pattern emerging when an obstacle seen by the right side sensor is only a persistent slight turn to left (around $0.3°$) as shown in Fig. 8. Finally, we have tested the robot behavior in more complex arenas obtaining good results (Fig. 9). In all the cases, the robot acquires the ability to reach the target with smooth trajectories.

Figure 9 about here.

As an example, Fig. 10 shows the sequence of patterns which have driven the navigation during one of these simulations.

IV. EXPERIMENTS AND RESULTS: ROVING ROBOT

A. Roving robot

Figure 10 about here.

The good results obtained in simulation opened the way to the application of the control algorithm in a real environment using a roving robot. In this section the complete framework and the experimental results are discussed. The experimental platform is constituted by a computer that implements the control architecture and the learning algorithms, a mobile robot and a wireless communication system.

Figure 11 about here.

The robot used for the experimental set-up is a Lynxmotion $4WD2$ Robot (see Fig. 11). It is a classic four wheel drive rover controlled through a differential drive system, the robot dimensions are $35\ cm \times 35\ cm$. The robot is equipped with 4 DC $50:1$ gear head motors and with an $RF04$ USB radio telemetry module for remote control operations and sensor acquisition. Moreover it is equipped with four infrared distance sensors $GP2D12$ from Sharp and with a digital compass $CMPS03$ for measuring the phase between the robot orientation and the direction to be followed. Two infrared sensors have been placed in the front side and the others in the left and right side at $45°$ with respect to the front orientation. We use the shortest between the two front side sensor outputs to activate the front side SN. The detection range of the infrared sensors is set from 14 to $64\ cm$ to limit noise. The ten steps, in which the x-axis of the SN activation functions is divided, correspond to $5\ cm$ each. Each sensor output, preprocessed by a PIC microcontroller, is transmitted to the PC. The same structure was already successfully used to investigate reactive navigation strategies (Arena et al., 2006b).

In this work, the task assigned to the robot is direction-following, where the target direction is established at the beginning of each trial and randomly changed during the experiment. During the navigation it also has to avoid the obstacles which appear in the sensory detection range. The hardware implementation corresponds to a basic navigation problem that can be already solved through other traditional algorithms (Borenstein & Koren, 1991) as well as new strategies based on Q-learning policy (Minato & Asada, 1998. Nevertheless our goal is to demonstrate the feasibility of the approach that is extremely general and can be applied to learn how to fulfil several different tasks specifying the robot mission inside the RFD and the type of actions that the robot can execute in the environment (e.g. moving, grasping, jumping, and so on). Furthermore the RF, in term of degree of satisfaction can be also modeled by the robot during the initial exploration of its working space on the basic reflex pathways associated to unconditioned stimuli, coming from contact sensors, low level target detector sensors and other (Arena et al., 2005, Arena et al., 2006a).

B. Learning phase

To use the control architecture on the real robot, we had to do only a minor modification to redefine in the RF the contributions related to the obstacle distances in terms of cm instead of $pixels$. Changing the task from target aiming to direction-following translated in the replacement of the robot-target direction with the direction to follow.

The learning phase is performed in an environment filled with black obstacles of various dimensions. It took about 160 minutes to create knowledge that makes the robot able to accomplish its own task in extremely different environmental situations. It is important to notice that, with this approach, all the robot knowledge is stored in the sensory neurons and in the pattern vector so the plasticity of the system is distributed among the sensory and the motor level as underlined by recent biological studies on insects.

C. Testing phase

A first test was carried out to demonstrate the ability to follow a given direction. Initially the angle between the robot orientation and the target direction was $80°$. The robot was driven by the pattern 12703 whose associated action is *turn to the left* ($w_{p,12703} = 8.6°$). The target direction was reached near a wall and, once the Right side obstacle sensor detected the wall, the pattern changed and the robot tends to turn further to the left. This results in a zig-zag behavior (Fig. 12).

Figure 12 about here.

The second test proves the ability to avoid an obstacle thanks to the learned capability of turning to the left of about $40°$ in presence of a frontal obstacle (Fig. 13): this is the result both of the learning phase and of the choice of a small $w_m = 7\ cm$, i.e. the module of the actions.

Figure 13 about here.

A third test was performed in order to demonstrate the ability to deal with a more complex environment by navigating under the guide of the Turing patterns acquired during the learning phase. Thus, in order to show the performance of the roving robot even in critical conditions, it was placed in a arena of dimensions $3m \times 3m$, filled with 3 obstacles in addition to the walls.

Figure 14 about here.

Fig. 14 shows the trajectory followed by the roving robot and the different Turing patterns which constitute the specific internal representations of the robot surroundings as learned during the learning phase.

Figure 15 about here.

Finally, Fig. 15 shows all the Turing patterns used in the previous trajectory, reporting for each pattern the number of occurrences and the associated action.

Videos are available on the web (Robot videos).

V. CONCLUSIONS

In this paper a new sensing-perception-action methodology is applied to the task of learning navigation. The methodology is presented and validated through simulations and experimental results on a roving robot. It is to be underlined that algorithms dedicated to face with the navigation task could even give better results: the potentiality of our approach lies in its generality. In fact the approach can be easily migrated to other applications, when the task is defined through the reward function, and the robot has to autonomously discover “how” to solve it. The approach, for example, is being actually applied to a more complex structure, an hexapod robot, where the control actions are much more complex, and the task of reaching a target could include not only avoiding obstacles by turning, but also climbing over steps, changing locomotion type and so on. In this case patterns indicate the particular scheme of leg motions, which should be applied in front of particular environment conditions. The above described framework is suitable to be included in a more complex and fine bio-inspired architecture aiming to emulate an insect brain at least from a functional point of view. A wider set of heterogeneous sensors such as microphones and a camera could be included. Moreover, a “Correlation layer” could be added, with the aim to learn specific temporal relations among sensory stimuli. This will allow relevant behaviors, like anticipation and cancelling stimuli reafference. The implementation of the whole architecture on board on the robot in view of a more complete and autonomous interaction with complex and cluttered environment is also envisaged.

VI. ACKNOWLEDGMENTS

The work was partially supported by the European Commission under the project FP6-2003-IST2-004690 SPARK and the MIUR-PRIN 2005 project: “Nonlinear dynamic network: analysis and application”. The authors also acknowledge the contribution of Mr. G. Paglia and Mr. R. Ranno in the simulation phase.

References

Adamatzky, A., Arena, P., Basile, A., Carmona-Galán, R., De Lacy Costello, B., Fortuna, L., Frasca, M., & Rodrguez-Vázquez, A. (2004). Reaction-Diffusion Navigation Robot Control: From Chemical to VLSI Analogic Processors. *IEEE Transactions On Circuits And Systems I, 51*, 926-938.

Arena, P., & Fortuna, L. (2002). Analog cellular locomotion control of hexapode robots. *IEEE Control System Magazine, 22(6)*, 21-36.

Arena, P., Fortuna, L., Frasca, M., & Sicurella, G. (2004). An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *IEEE Transactions on Systems, Man and Cybernetics, Part B, 34(4)*, 1823-1837.

Arena, P., Fortuna, L., Frasca, M., Lombardo, D. & Patan, L. (2005). Learning efference in CNNs for perception-based navigation control, *International Symposium on Nonlinear Theory and its Applications (NOLTA)*, Bruges, Belgium.

Arena, P., Fortuna, L., Frasca, M., Patané, L., & Pavone, M. (2006a). Towards Autonomous Adaptive Behavior in a bio-Inspired CNN-Controlled Robot. *IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, Island of Kos, Grecia.

Arena, P., De Fiore, S., Fortuna, L., Frasca, M., Patané, L., & Vagliasindi, G. (2006b). *Weak Chaos Control for Action-Oriented Perception: Real Time Implementation via FPGA*. Proceedings of International conference on Biomedical Robotics and Biomechatronics.

Arena, P., Crucitti, P., Fortuna, L., Frasca, M., Lombardo, D., & Patané, L. (2007a). Turing patterns in RD-CNNs for the emergence of perceptual states in roving robots, *International Journal of Bifurcation and Chaos, 17(1)*, 107-127.

Arena, P., Cruse, H., Fortuna, L., Lombardo, D., Patané, L., & Rapisarda, R. (2007b). Adaptive bio-inspired landmark identification for navigation control. *Proceedings of the SPIE conference: Microtechnologies for the New Millennium 2007*.

Arkin, R. C. (1997). *Behaviour Based Robotics*, Cambridge, MA:MIT Press.

Borenstein, J., & Koren, Y. (1991). The vector field histogram fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation, 7*, 278-288.

Brooks, R. A. (1994). Intelligence without reason. *Proceedings of 12th International Joint Conference on Artificial Intelligence, eds. J. Mylopoulos & R. Reiter, San Mateo, CA:Morgan Kaufmann.*

L.O.Chua, M.Hasler, G.S.Moschytz, J.Neirynck, 1995, ”Autonomous Cellular Neural Networks: A Unified Paradigm for Pattern Formation and Active Wave Propagation”, *IEEE Trans. on Circuits and Systems - Part I, 42, 10*.

Franz, M. O., & Mallot, H. A. (2000). Biomimetic robot navigation, *Robotics and Autonomous Systems, 30*, 133-153.

Freeman, W. J. (2004). How and why brains create meaning from sensory information. *International Journal of Bifurcation and Chaos, 14*, 515-530.

Goras, L., & Chua, L. (1995). Turing Patterns in CNNs - Part I, II. *IEEE Trans. Circuits and Systems - I, 42*, 602-626.

Kelso, J. A. S. (1995). *Dynamic patterns: The self-organisation of brain and behavior*, Cambridge MA: MIT press.

Manganaro, G., Arena, P., & Fortuna, L. (1999). *Cellular Neural Networks: Chaos, Complexity and VLSI processing*, New York:Springer-Verlag.

Minato, T., & Asada, M. (1998) Environmental Change Adaptation for Mobile Robot Navigation. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1998 (IROS ’98), pp.1859-1864*.

Murray, J. D. (2002). *Mathematical Biology I: An Introduction (third edition)*, New York:Springer-Verlag.

Nolfi, S. (2002). Power and Limits of Reactive Agents, *Neurocomputing, 42(1)*, 119-145(27).

Orlovsky, G. N., Deliagina, T. G., & Grillner, S. (1999). *Neural control of locomotion. From mollusc to man*, Oxford: University press.

Pavlov, I. I. (1927), *Conditioned Reflexes*, London:Oxford University Press.

Schul, J. L. (1998), Song recognition by temporal cues in a group of closely related bushcricket species (genus Tettigonia), *Journal of Comparative Physiology A, 183*, 401-410.

Schulten, K. (2002), *Theoretical Biophysics of Living Systems*, available in
http://www.ks.uiuc.edu/Services/Class/PHYS498TBP/spring2002/neuro_book.html.

Skinner, B. F. (1974). *About behaviorism*, New York:Alfred Knopf.

Tani, J., & Fukumura, N. (1994a). Embedding task-based behavior into internal sensory-based attractor dynamics in navigation of a mobile robot. *Proceedings of the IEEE Int. Conf. of Intelligent Robot and Systems*, 886-893.

Tani, J., & Fukumura, N. (1994b). Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks, 7(3)*, 553-563.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Phil. Trans. Roy. Soc. Lond. B, 237*, 37-72.

Verschure, P. F. M. J., Voegtlin, T., & Douglas, R. J. (2003). Environmentally mediated synergy between perception and behaviour in mobile robots. *Nature, 425*, 620-624.

Webb, B., & Scutt, T. (2000). A simple latency dependent spiking neuron model of cricket phonotaxis. *Biological Cybernetics, 82(3)*, 247-269.

Webpage available at www.spark.diees.unict.it/AdaptiveBehavior.html.

Paolo Arena Paolo Arena received the degree in Electronic Engineering and the Ph.D. in Electrical Engineering in 1990 and in 1994, respectively, from The University of Catania, Italy. He is currently Associate Professor of System Theory. He published more than 130 technical papers, six books and several international patents. His research interests include adaptive and learning systems, nonlinear systems, neural networks, cellular neural networks, collective behavior in living and artificial neural structures and cognitive systems. He is a Senior Member of the IEEE and served as an Associate Editor of the IEEE Transactions on Circuits and Systems, Part I.

Luigi Fortuna Luigi Fortuna is Full Professor of System Theory at the University of Catania since 1994. He has published more than 300 technical papers ad is co-author of seven books amongst which Cellular Neural Networks (Springer 1999). He holds several USA patents. His scientific interests include Nonlinear Science and Complexity, Chaos, Cellular Neural Networks with Applications in Bioengineering. Since 2000 he is IEEE Fellow.

Davide Lombardo Davide Lombardo was born in Catania, Italy, in 1979. He graduated in Computer Science Engineering in 2004 and enter the Ph.D. course of Electronics and Automation Engineering in 2005 at the University of Catania, Italy. His scientific interests include Biomedical engineering, Complex Systems, bio-inspired robotics and perception.

Luca Patané Luca Patané was born in Catania, Italy, in 1978. He graduated in Computer Science Engineering in 2001 and received the Ph. D. in Electronics and Automation Engineering in 2005, at the University of Catania, Italy. Currently, he is research associate at the University of Catania. His scientific interests include Cellular Neural Networks, Complex Systems, bio-inspired robotics and perception. Recently his research activity is focused on the formalization of an insect brain computational model.

LIST OF FIGURES

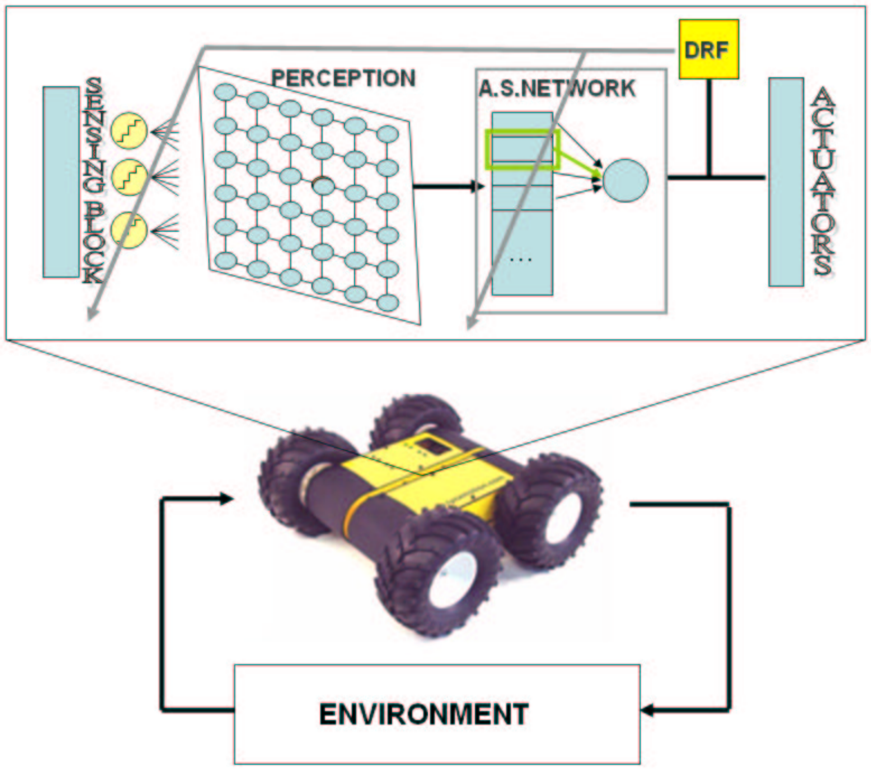
SENSING BLOCK
PERCEPTION
A.S.NETWORK
DRF
ACTUATORS
...
ENVIRONMENT

Fig. 1.

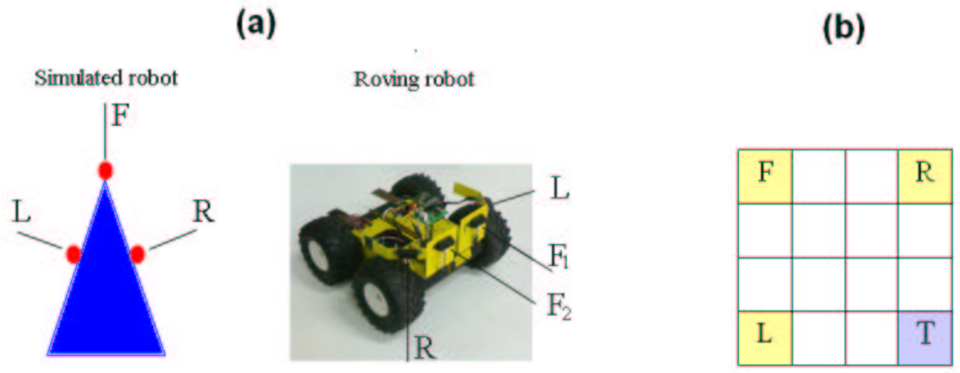
(a)
Simulated robot
F
L
R
Roving robot
L
F1
F2
R
(b)
F
R
L
T

Fig. 2.

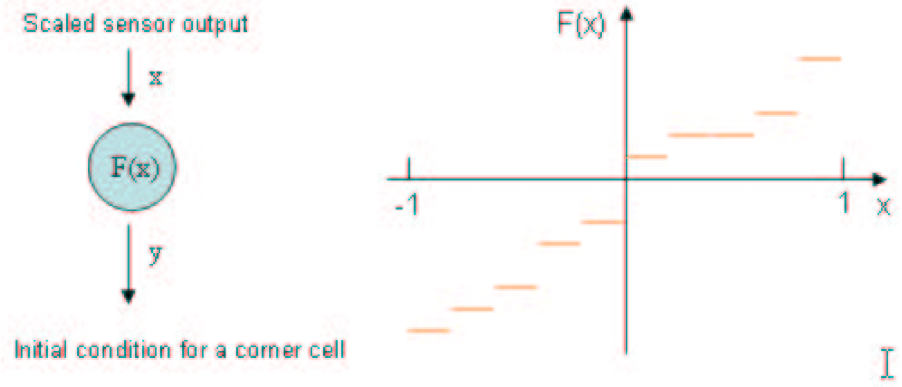
Scaled sensor output
x
F(x)
y
Initial condition for a corner cell
F(x)
-1
1
x

Fig. 3.

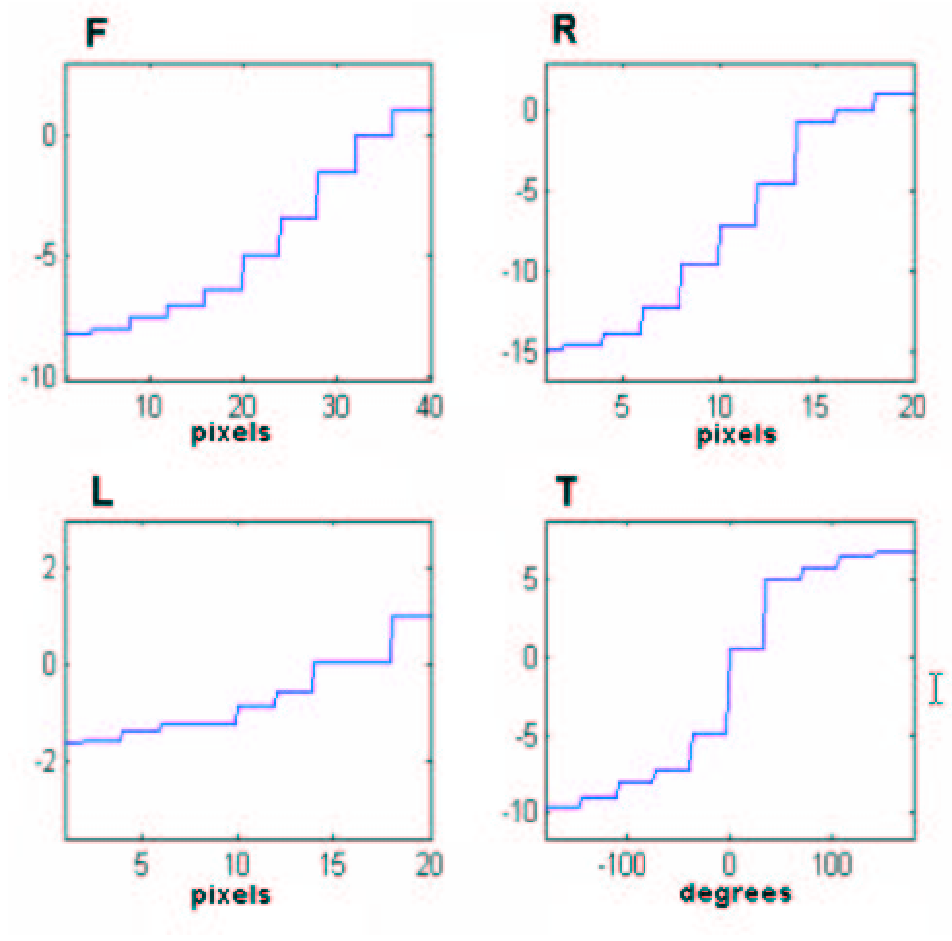
F
R
L
T
pixels
pixels
pixels
degrees
0
-5
-10
10
20
30
40
-15
5
15
2
-2
-100
100

Fig. 4.

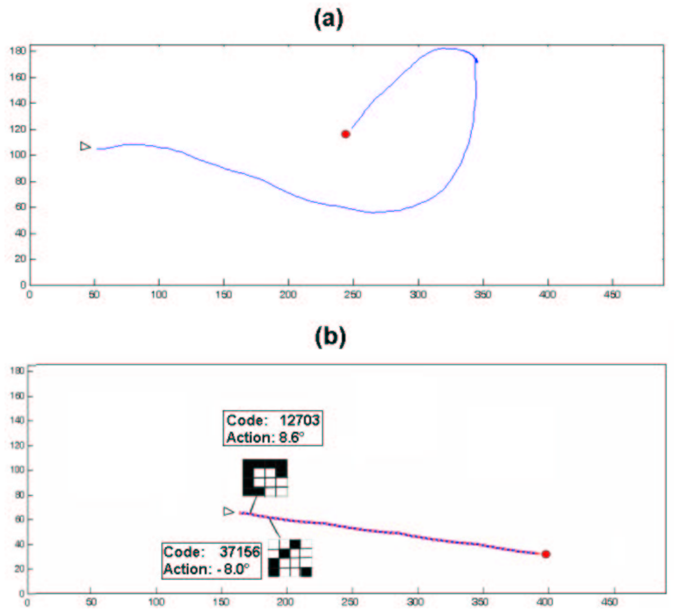
(a)
(b)
Code: 12703
Action: 8.6°
Code: 37156
Action: -8.0°

Fig. 5.

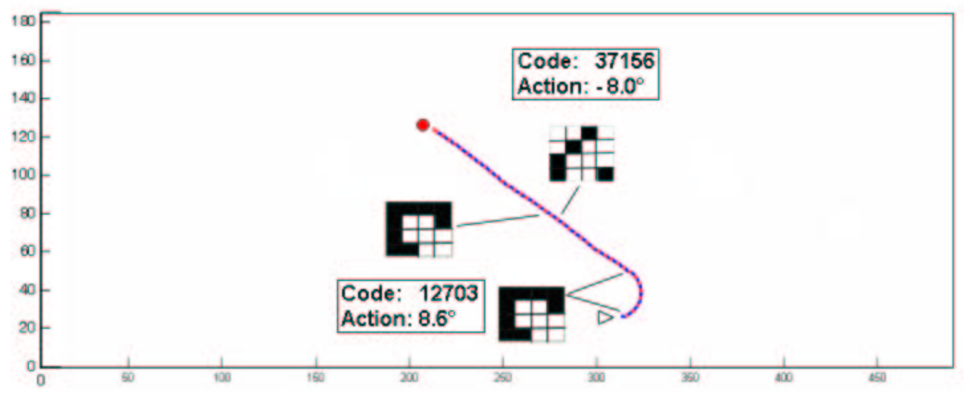
Code: 37156
Action: -8.0°
Code: 12703
Action: 8.6°

Fig. 6.

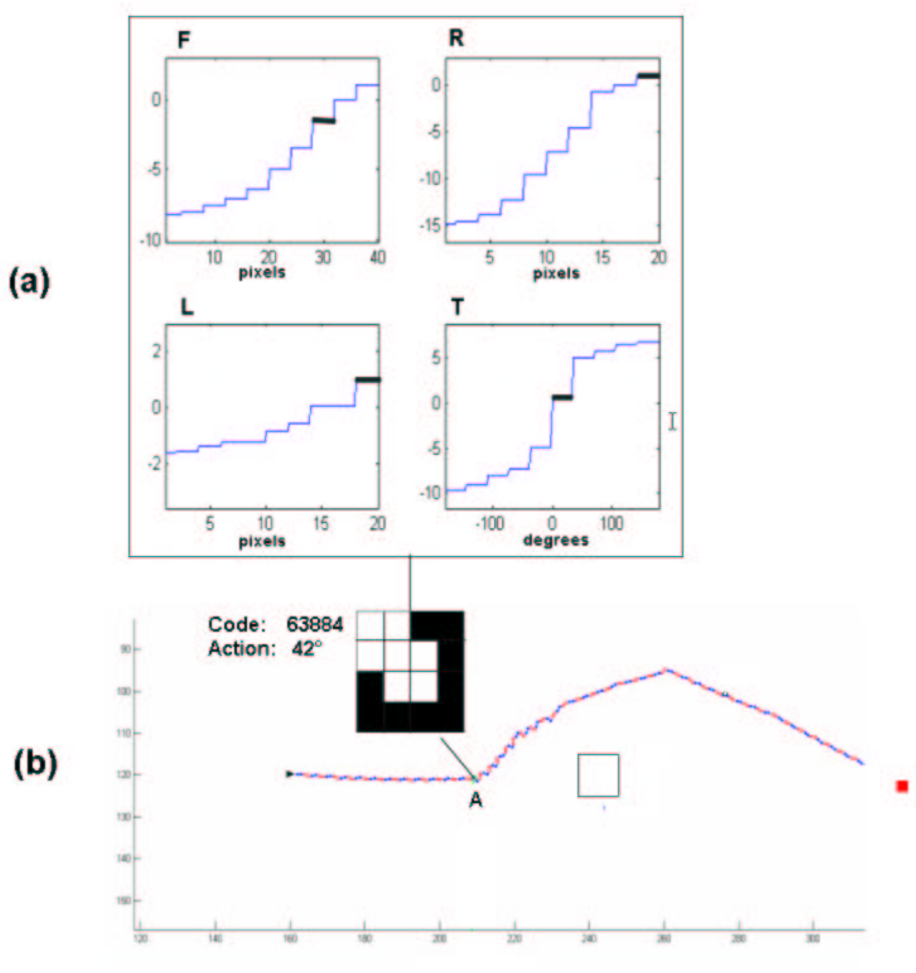

Fig. 7.

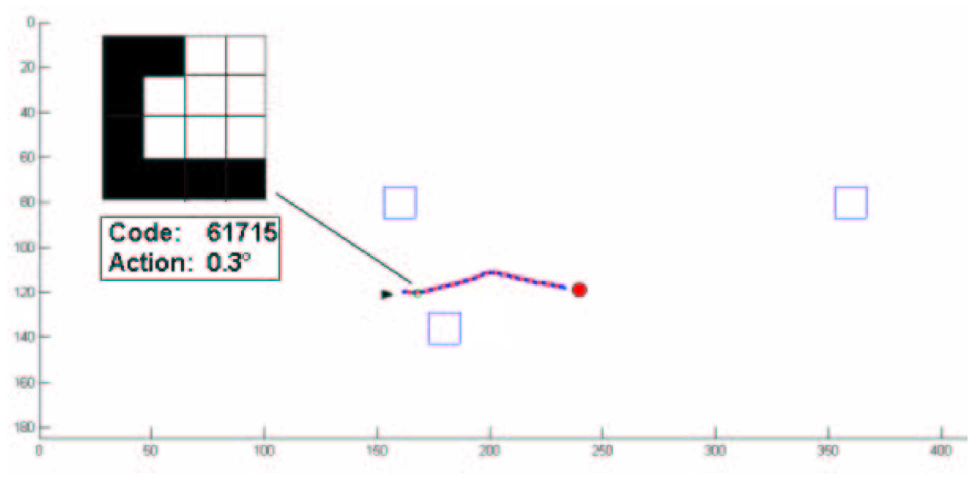
Code: 61715
Action: 0.3°
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200
250
300
350
400

Fig. 8.

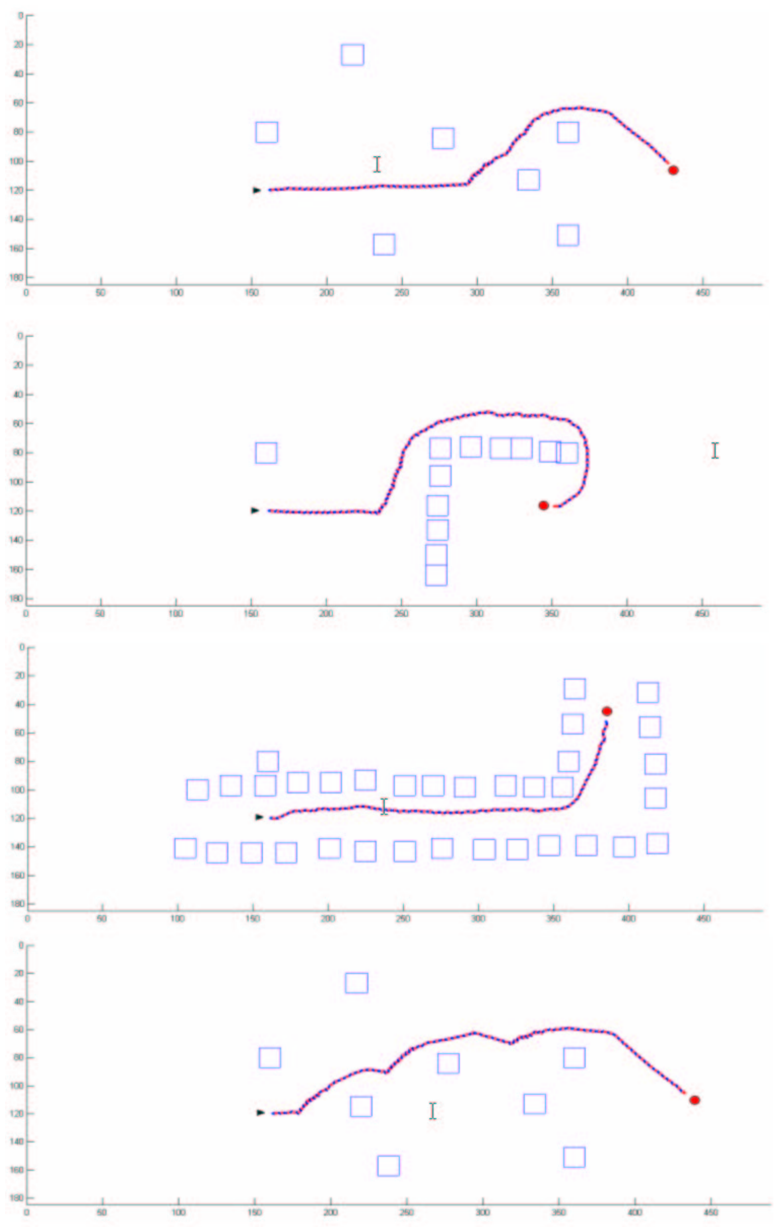
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200
250
300
350
400
450

Fig. 9.

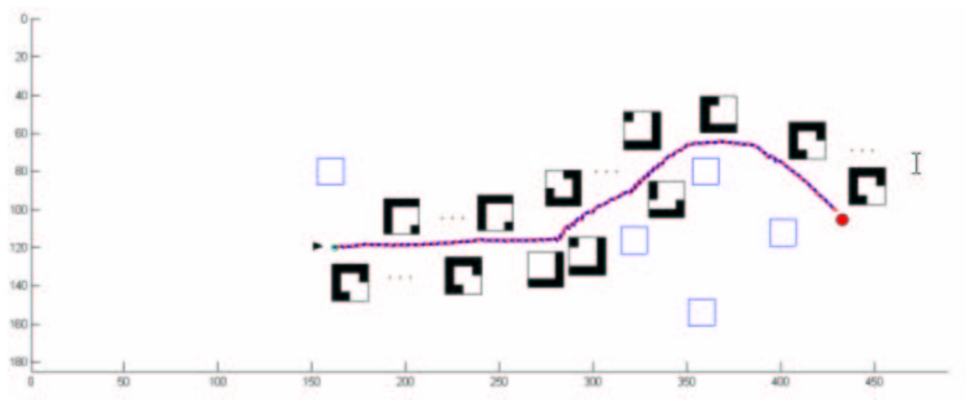
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200
250
300
350
400
450

Fig. 10.

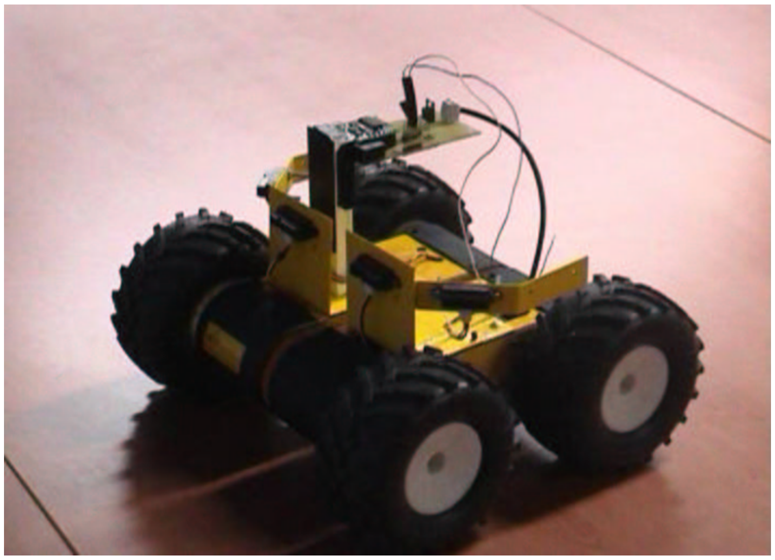

Fig. 11.

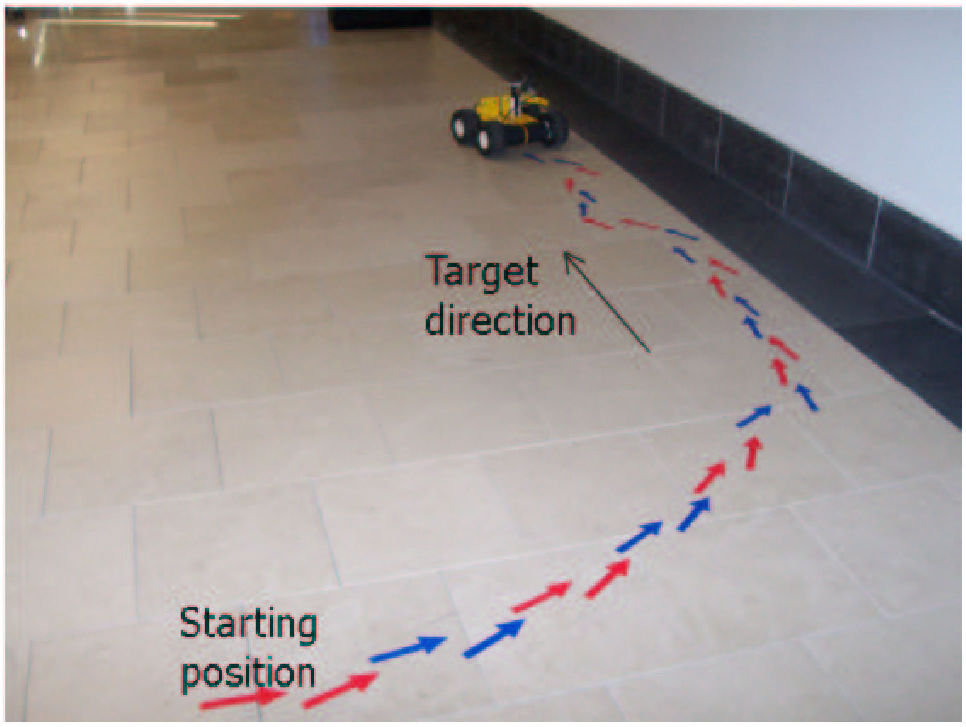

Fig. 12.

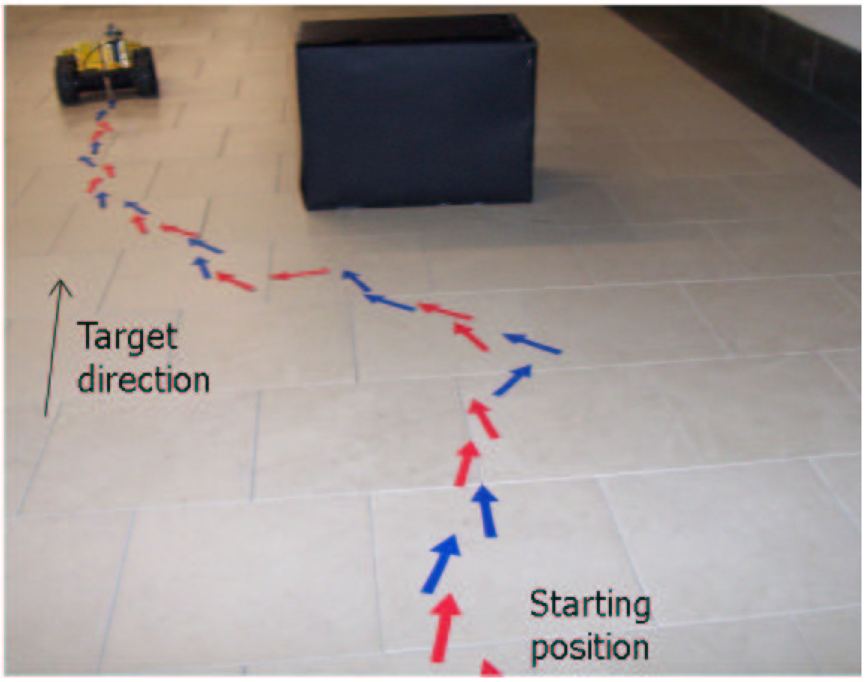

Fig. 13. [

Trajectory followed in heading the target direction and avoiding an obstacle.]

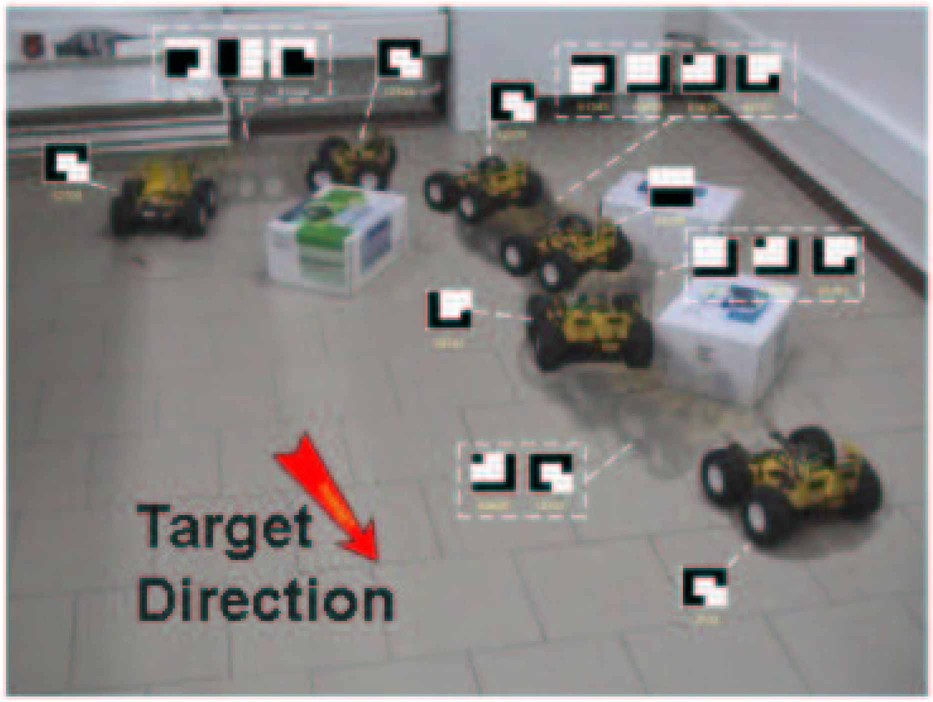

Fig. 14.

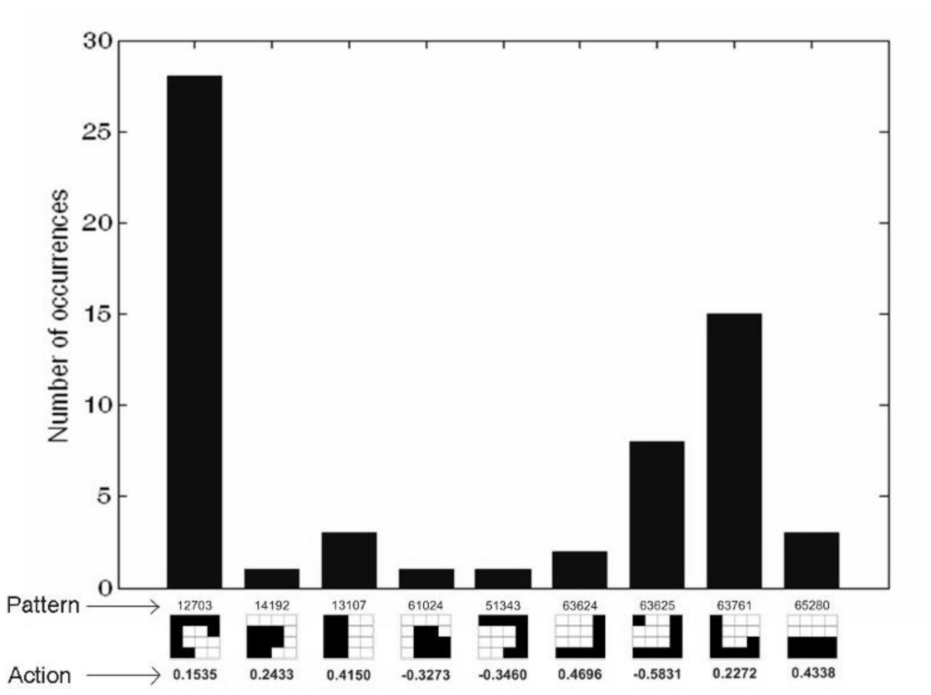
Number of occurrences
30
25
20
15
10
5
0
Pattern
12703
14192
13107
61024
51343
63624
63625
63761
65280
Action
0.1535
0.2433
0.4150
-0.3273
-0.3460
0.4696
-0.5831
0.2272
0.4338

Fig. 15.