Cellular Non-linear Networks as a New Paradigm for Evolutionary Robotics

Eleonora Bilotta and Pietro Pantano
Università della Calabria
Italy

1. Introduction

One of the most active fields of research in evolutionary robotics is the development of autonomous robots with the ability to interact with the physical world and to communicate with each other, in "robot societies". Interactions may involve a range of different motor actions, motivational forces and cognitive processes. Actions, in turn directly affect the agent's perceptions of the world. In the "Action/Perception-Cycle" (see Figure 1), biological organisms are integrated sensorimotor systems. This means that intelligent processes require a body, and that symbols are grounded in the environment in which animals live (Harnad, 1990). In short, behavior is fundamentally linked to cognition. This is true for humans, animals and artificial agents. Without this grounding, artificial animals and agents cannot live and behave successfully in their artificial environments. One way of achieving it, is to use Genetic Algorithms to evolve agents' neural architecture (Nolfi & Floreano, 2000). This creates the prospect of robots that can live in complex socially organized communities in which they communicate with humans and with each other (Cangelosi e Parisi, 2002). According to these authors, cognition is an intrinsically embedded phenomenon in which the dynamical relations between the neural system, the body and the environment play a central role. In this view, agents are dynamical systems and cognitive functioning has to be understood using tools from dynamical system theory (van Gelder, 1995, 1998a; 1998b; Bilotta et al., 2007a-2007f). This perspective on cognition has been called the Dynamical and Embodied view of Cognition (DEC) (Keijzer, 2002).
In this chapter we describe our own contribution to Evolutionary Robotics, namely a proposal for a new generation of believable agents capable of life-like intelligent communicative and emotional behavior. We focus on CNNs (Cellular Neural Networks) and on the use of these networks as robot controllers. In previous work, we used Genetic Algorithms (GAs) to evolve Artificial Non-linear Networks (ANNs) displaying artificial adaptive behavior, with features similar to those observed in animals and humans. In (Bilotta et al. 2006), we replaced ANNs with a new class of dynamical system called Cellular Non-linear Networks (CNNs) and used CNNs to implement a multilayer locomotion model for six-legged artificial robots in a virtual environment with many of the characteristics of a physical environment. First invented by Chua and co-workers (1988), CNNs have been extended to create a CNN Universal Machine (CNN-UM) (Roska & Chua, 1993), the first algorithmically programmable analog computer chip suitable for the modeling of sensory-

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

motor processes. Applications of the chip include the modeling of the mammalian retina (Roska et al. , 2006) and motor coordination in life-like robots (Arena & Fortuna, 2002). CNNs can be organized in complex architectures of one, two or three-dimensional processor arrays in which the cells are identical non-linear dynamical systems, whose only connections are local. Systems composed of CNNs share a large number of features with living organisms: local connectivity and activity, nonlinearity and delayed synapses for processing tasks, the role of feedback in influencing behavior, as well as combined analog and logical signal-processing. Direct implementation on silicon provides robust, economic, fast and powerful computation. Thousands of experiments have demonstrated the possibility of digitally programming analog dynamics. This has made the CNN paradigm into a useful tool for robot applications.

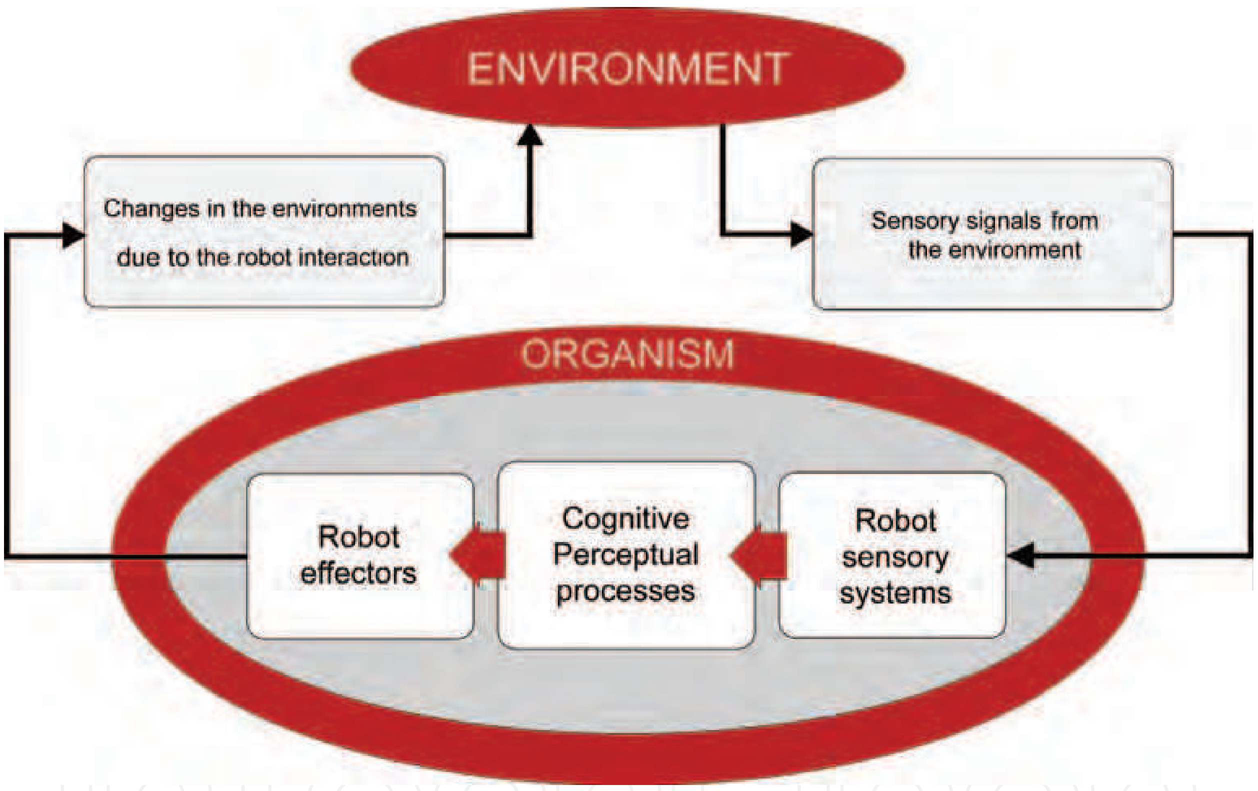

ENVIRONMENT
Changes in the environments due to the robot interaction
Sensory signals from the environment
ORGANISM
Robot effectors
Cognitive Perceptual processes
Robot sensory systems

Figure 1. The Action/Perception-Cycle

In this chapter we will proceed as follows. In Section 2, we introduce the concept of Cellular Non-linear Networks as innovative dynamical robot controllers that can be implemented both in simulation and as hardware prototypes. In section 3 we present CNN architectures for different cognitive and motor processes (CNN computing: visual and motor modalities); in section 4 we present the RoVEn simulation environment – an environment specifically developed for the evolution of CNN-based robot controllers. In Section 5 we describe our experiments in simulated environments. Section 6 describes the implementation of prototype chips which can act as behavioral modules for physical robots. In the conclusions, we summarize the evidence that the CNN paradigm can dramatically improve current research in Evolutionary Robotics.

2. Cellular Neural Networks

CNNs (Cellular Neural Networks) were first introduced in 1988 by Leon Chua and Yang (Chua & Yang, 1988). They are dynamical systems. For this reason they are sometimes referred to as Cellular Nonlinear Networks. CNNs have applications in many domains from image recognition to robot control. Given their ductility, the ease with which they can be implemented and the dynamics they display they can be considered a *paradigm for complexity*. CNNs can be organized in one- two- or three dimensional topologies (see Figure 2). As we will see in the following sections, CNN applications are of relevance to many different disciplines including Robotics, Dynamic Systems Theory, Neuro-psychology, Biology and Information Processing. One of the first applications was *image processing*. A digitalized image can be represented as a two dimensional matrix of pixels. To process it with a CNN, all that is necessary is to use the normalized colors of pixels (i,j) as the initial state of the network. The network then behaves as a non-linear dynamical system with a number of equations equal to the number of cells. The network makes it possible to perform a number of useful operations on the image. These include edge detection, generation of the inverse figure etc. For additional information on this topic and on other applications of CNNs, see (Chua, 1998).
As we will see in Section 6, CNNs are very similar to programmable non-linear dynamical circuits – and in fact physical implementations often use these circuits. Given CNN's non-linear design, CNNs often produce *chaotic dynamics.* Given the presence of *local activity* in individual cells, it is possible to observe a broad range of emergent behaviors. One of these is the formation of Turing patterns (Chua, 1995), which are often used in robot control. (Arena et al., 1998; Arena & Fortuna 2002).
As with all complex emergent phenomena, it is difficult to identify the full range of non-linear dynamic behaviors a CNN can produce and equally hard to control the network's behavior. The main reason is that the dynamics of individual cells are controlled by first order non-linear differential equations. Given that the cells are coupled, the equations are also coupled. This makes them similar to the Lorenz system and Chua's circuit (Bilotta et al. 2007a-2007f) which also display highly complex, mathematically intractable dynamics. What is special about CNNs is that they can be used to reproduce the complex dynamics of other non-linear systems such as Chua's circuit (Bilotta et al., 2007a). In this sense, we can consider CNNs as a *general model* or a *meta-model* for other dynamical systems.
Another important application of CNN is in the numerical solution of Partial Differential Equations (PDEs). If we use a grid to create a discretized space, in which variable values are represented by intersections on the grid, the derivatives with respect to spatial variables can also be discretized while the derivatives with respect to time remain unchanged. These discretized differential equations can be mapped onto the equations regulating the behavior of the CNN. In this way, CNNs can simulate a broad range of physical phenomena. For more information and a review of this aspect of CNN see (Chua, 1998).
Specific CNN architectures can reproduce a broad range of *non-linear phenomena* that biologists, neurologists and physics have observed in active non-linear media and in living tissue. These include solitons, eigen waves, spiral waves, simple patterns, Turing patterns etc. Given CNN's local connectivity, diffusion is a natural property of the network; diffusion-reaction dynamics can be simulated using the interactions between an inhibitory and an excitory layer. This class of two-layer CNN has been called a Reaction-Diffusion CNN.

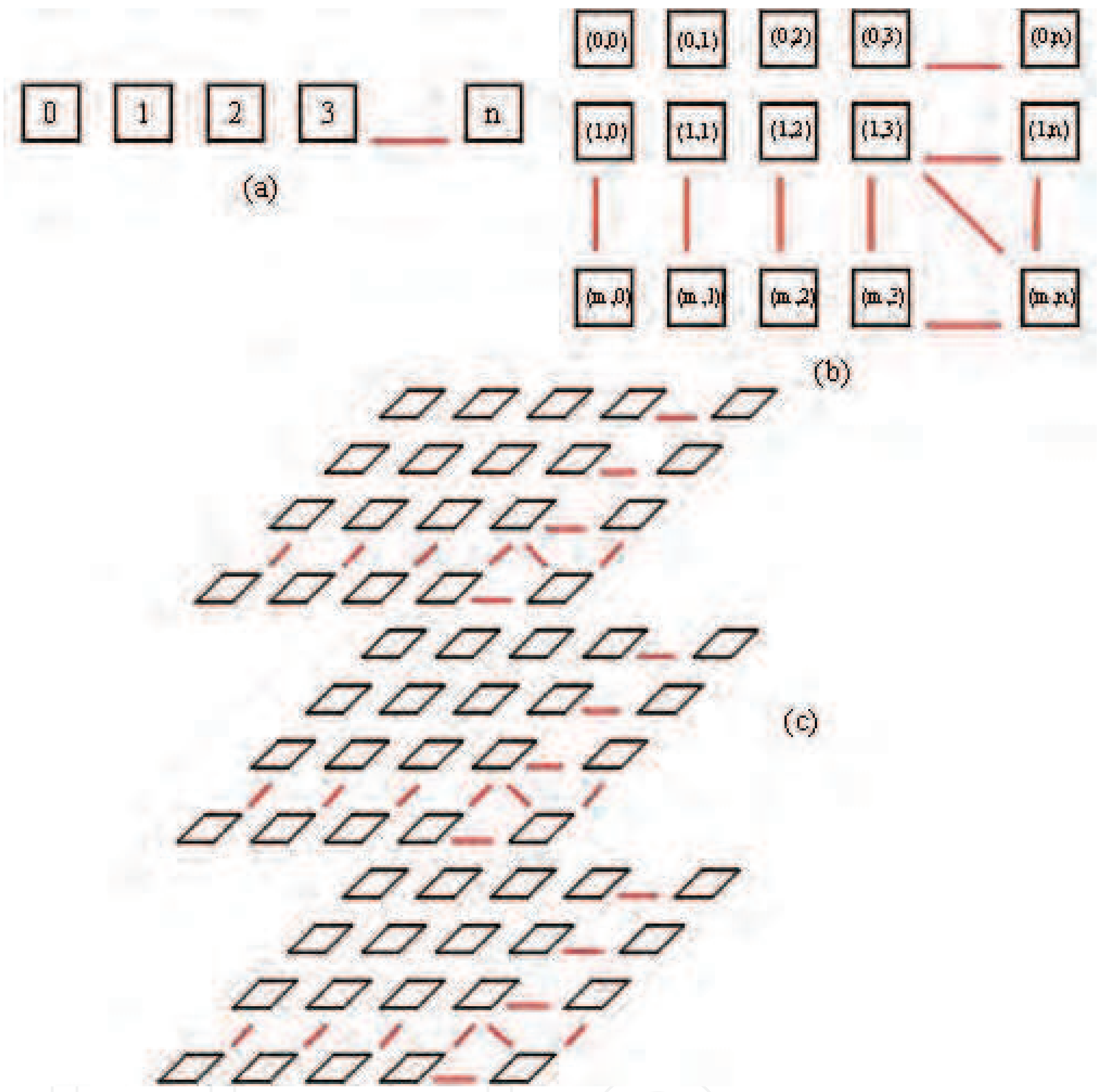


Figure 2. CNN topologies a) A linear topology; b) A two-dimensional topology; c) A three-dimensional topology

From a mathematical point of view a CNN is a discrete set of continuous dynamical variables called "cells". Each cell is associated with three independent variables: the input, the threshold and the initial state. The cell's dynamics are influenced by close-by cells $S_{ij}(r)$ in a neighborhood of radius r. Thus, if we consider a CNN with dimensions LxM, the dynamics of cell C_{ij}, located on the i-th row and the j-th column, are influenced by cells in the neighborhood $S_{ij}(r)$ defined as:

$$S_{ij}(r) = \{C_{ij}, with 1 \le i \le L; 1 \le j \le M\} \quad (1)$$

that is the set of cells lying within a sphere of radius r, centered on C_{ij}.
The standard equation for CNNs (Chua, 1988) is thus:

$$\dot{x}_{ij}(t) = -x_{ij} + \sum_{kl \in S_{ij}(r)} a_{kl} y_{kl} + \sum_{kl \in S_{ij}(r)} b_{kl} u_{kl} + z_{ij}$$

$$y_{ij}(t) = f(x_{ij}) = \frac{1}{2}\left(|x_{ij}+1| - |x_{ij}-1|\right)$$

$$i = 1,2,\ldots, L, j = 1,2\ldots, M$$

where x_{ij}, y_{ij}, u_{ij} and z_{ij} are the scalar functions *state*, *output*, *input* and *threshold* for cell C_{ij}; a_{kl}, and b_{kl} are additional scalar functions which we will refer to as *synaptic weights*.
A CNN with r=1, can be identified by just 19 real numbers (a uniform *threshold* $z_{ij} = z$, 9 synaptic weights for feedback a_{kl}, and 9 synaptic weights for control b_{kl}). These 19 numbers are the CNN's "genes". The set of all CNN genes constitute the CNN *genome*.

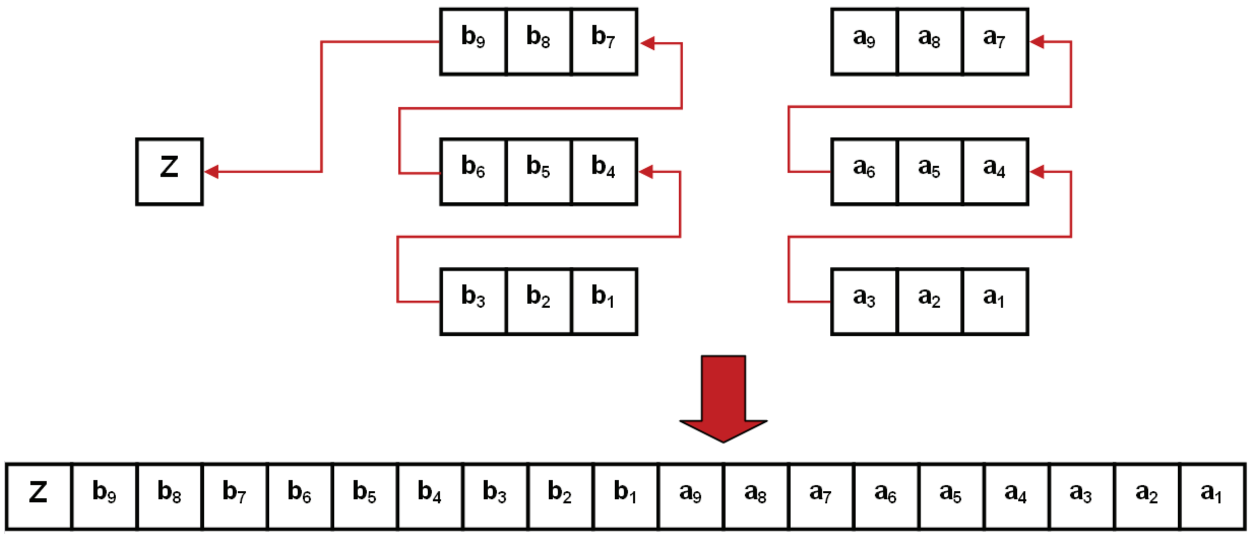


Figure 3. Constructing the genes of a CNN

As already mentioned, there are many applications where the most useful CNN is the network shown in Figure 4. In *two-layer* CNNs each cell is double. Alternatively we could say that each cell has two state variables. We can thus distinguish between C^1_{ij} (cells in layer 1) and C^2_{ij} .(cells in layer 2). In this kind of two layer network the neighborhood of a cell is defined by:

$$N_{r1}(i,j) = \{C^1{}_{ij}, with 1 \le i \le L; 1 \le j \le M\}$$

$$N_{r2}(i,j) = \{C^2{}_{ij}, with 1 \le i \le L; 1 \le j \le M\}$$

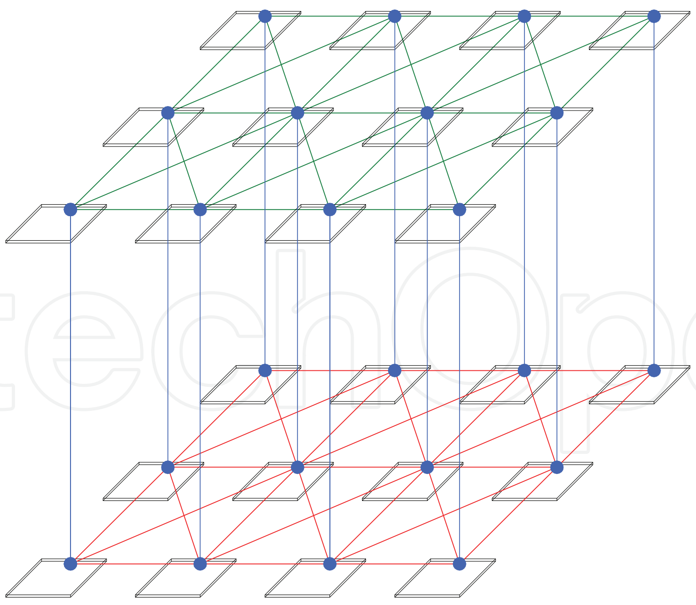


Figure 4. An example of a two layer planar architecture showing connections between cells in the same layer (green in layer 1, red in layer 2) and between twin cells in different layers (shown in blue)

Thus we obtain one definition of neighborhood for cells belonging to layer 1 and a second definition for cells belonging to layer 2. Cells intercommunicate only if they have the same index.

3. CNNs Architectures

We have already seen that CNNs can be useful in a broad range of applications. Now we will focus on three specific applications in robotics:

a) Models of the Central Pattern Generator (CPG);
b) Applications in artificial vision and new models of artificial retinas.

3.1 Central Pattern Generator

One of the central problems in bio-inspired robotics is to reproduce animal locomotion in artificial settings. Experiments with simple invertebrates have helped researchers to understand and model the anatomical and functional mechanisms underlying animal locomotion (Arena & Fortuna, 2002; Schilling et al., 2007). In animals with very simple locomotion (such as certain worms and molluscs), movement depends on direct propagation of a signal through nerves. The soft structure of the animal's body allows it to synchronize with the waves and to produce a wave-like locomotion. This kind of phenomenon can be easily described by the reaction-diffusion PDEs mentioned earlier (Murray, 1989). As already stated, the solutions to the equations include autonomous oscillations with all the properties of an eigen-wave (Krinsky,1984). The same mechanism – in which continuous motion is maintained by an eigen-wave – is also present in higher animals. However, the underlying neural structure is much more complex and shows a much higher degree of organization. To model these behaviors mathematically we can begin by studying the various components of the animal's nervous system. The resulting model is built around a Central Pattern Generator (CPG) in which motor neurons are activated by

stimuli from the Central Nervous System (CNS). The neural structure of the CPG is very complex. As a result, it can do more than just generate the impulses controlling movement; it can also control the transition from one kind of movement to another (Pearson, 1993).
In one possible model (see Figure 5) the CPG is described as a complex, hierarchically organized excitory-inhibitory system. Within this system a group of control neurons (CNs) receives stimuli from the Central Nervous System and activates other neurons (LPGN) that generate timing signals appropriate to the desired form of locomotion (Calabrese, 1995).

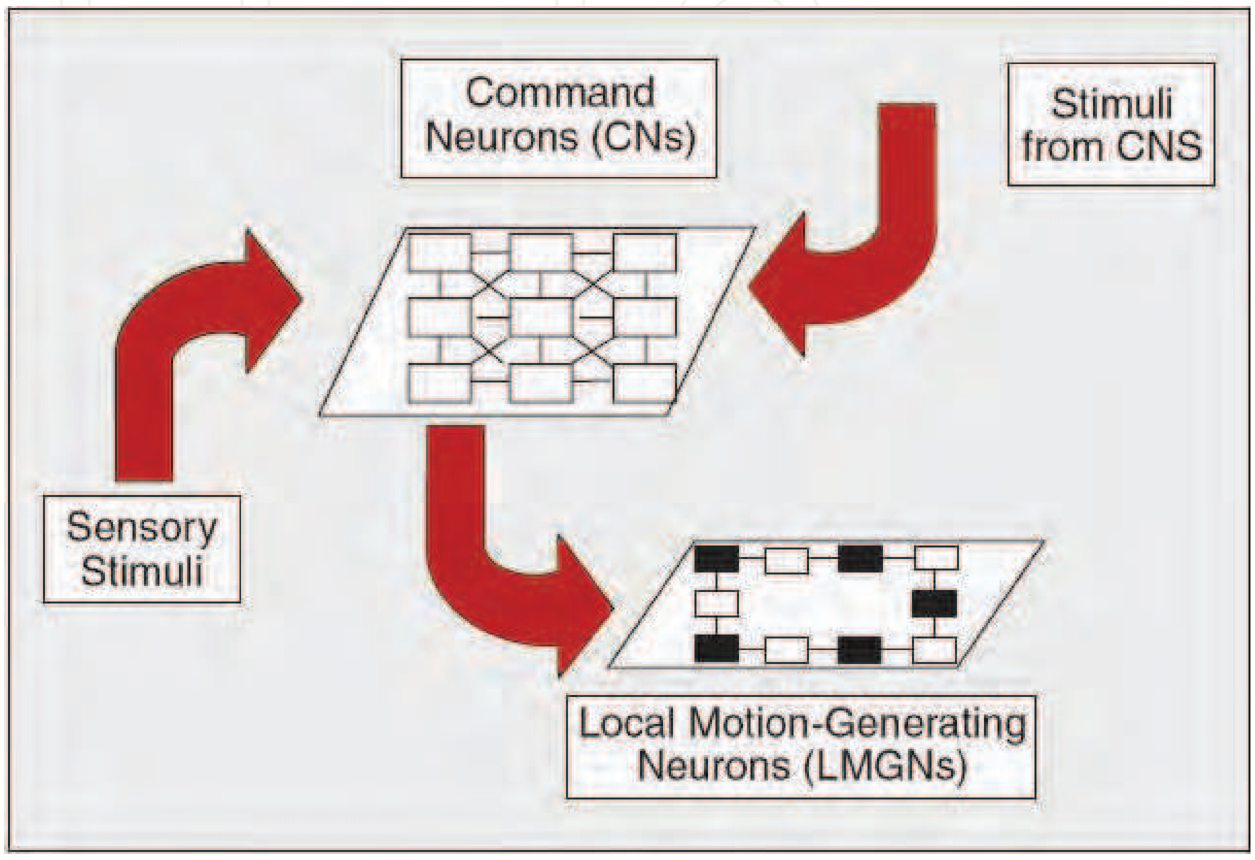


Figure 5. A schematic diagram of the CPG from (Arena & Fortuna, 2002)

This model can be easily represented by a two layer CNN, based on the reaction-diffusion equations mentioned earlier. The use of these equations allows it to generate spatial-temporal patterns such as eigen-waves and Turing patterns. This means we can use CNN both to model the CPG and to control robot locomotion. Further details are available in (Arena & Fortuna, 2002; Schilling et al., 2007).

3.2 Vision and Artificial Retinas

CNNs can be used to simulate higher level cognitive processes and are beginning to be used as models of complex processes relevant to the construction of artificial organs. One particularly important example is the retina. In a series of experiments in rabbits, Werblin and coworkers (Fried et al., 2005)-(Roska et al. 2006), have demonstrated that the retina pre-processes visual signals before sending them to the brain. More specifically, they show that before being sent to the cortex for final processing the retina uses excitory and inhibitory mechanisms to break it down into 12 separate sub-images. These are processed in parallel each with its own clock. The result is a seamless flow of visual information. On the basis of

this work, the authors went so far as to propose the idea of an *alphabet for vision* – and argued that understanding this natural language was one of the most important problems in modern science. It is a problem with great relevance to the construction of artificial organs.
As Werblin and Roska pointed out (Roska et al., 2006), it is extremely difficult to connect electrodes to individual retinal cells in the living rabbit. This makes it hard to understand how the rabbit responds to visual stimuli e.g. a one second flash or a minute's exposure to a natural scene. For their experiments and the analysis of the resulting data, they used a model based on CNNs. The results were astonishing and showed that a CNN can reproduce many phenomena observed in vivo. Obviously this modeling effort is still in its initial stages. Nonetheless it provides evidence that CNNs can be very useful in modeling complex natural phenomena such as those observed in the retina. Remember that CNNs are based on PDEs. With their emergent data processing properties, their parallel processing capabilities and their ability to process continuous flows of information they represent an important paradigm in modern complexity science.

4. Evolving CNNs with GAs

In the previous section we saw how we can use CNNs to model a range of processes (e.g. locomotion control, retinal image processing) that are highly relevant to robotics. This suggests they could act as a unifying model for a new generation of bio-inspired autonomous robots. The basic schema shown in Figure 6 can be used as a basic design both for physical and simulated robots.
To implement the schema, we intend to develop experimental scenarios in which simulated robots are controlled by dynamical systems which allow them to achieve the kinds of perceptual-motor, communication and emotional behavior they need to interact effectively with other robots and with humans. In these scenarios, our robots will demonstrate a high degree of autonomy and self-awareness.
It is obvious that implementing the architecture just described is a highly complex task which must necessarily be decomposed into subtasks. These include:
a. implementation of a sensory visuomotor architecture, allowing them to merge information in different sensory modalities into a coherent representation of the environment. By using CNNs, we intend to endow robots with artificial visual, auditory and haptic systems allowing them to recognize and categorize faces, objects, and scenes under varying viewing and dynamically changing environmental conditions (Roska & Chua, 1993; Gacsádi et al., 2006; Gacsádi and Szolgay, 2004);
b. implementation of an emotional architecture, organizing the robot's behavior. An emotional robot would use its physical actuators and cognitive skills to adapt to changing environmental conditions in real-time. In this setting, emotions trigger behavior. More specifically, moving objects, the postures/gestures of other robots, and sounds or facial expressions produced by human beings all generate different emotional states;
c. implementation of a communication system based both on non-verbal communication (Gesture and movement recognition, Face recognition) and a natural language model. Specific learning mechanisms will allow CNN- based devices to acquire, from human users the speech configurations of different natural languages A specific CNN-based architecture, combined with sensors attached to a human speaker, will allow the system to capture the emotions expressed by human subjects when interacting with a virtual interface (Face and emotion recognition). CNNs will perform recognition tasks in two sensorial modalities (auditory and

visual). We will then use GAs to spontaneously evolve spoken language. In parallel with this work, we will implement a robot controller that will enhance the cognitive mechanisms the robot uses to process emotions thereby dramatically improving human-computer interaction;
d. implementation emergent social behavior with hundreds of agents. We will use acoustic stimuli (known and unknown sounds) and visual stimuli to verify how agents' interactions with the external world influences the dynamics of large groups of robots and the way communication emerges within the group.

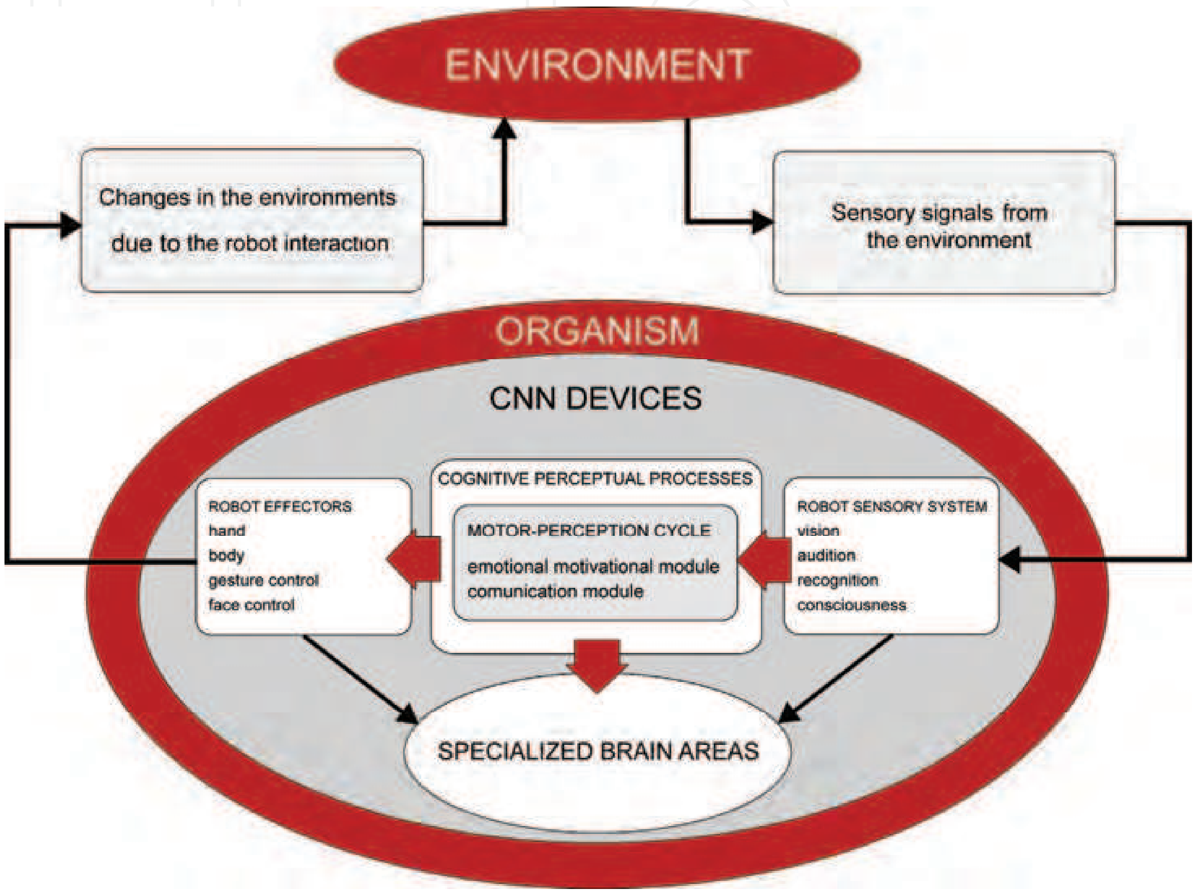


Figure 6. A cognitive architecture for CNN based Robots

The key features of our approach include:

1. Behavioral/cognitive modeling for robotics.
2. A dynamical approach to the modeling of robot cognition and emergent behavior.
3. Integration of mathematical models on different scales and at different levels of the physical functional architecture.
4. Robust behavioral hardware and software implementation with Cellular Neural Networks (CNNs), supporting parallel multidimensional processes that allow robots to robustly handle their interaction with the environment.
5. Robust implementation of robot controllers using Cellular Neural Networks (CNNs), evolved by evolutionary techniques (GAs).

It is obvious that what we are proposing will require a great deal of work and that at the moment we are only in the very early stages. So far we have developed an initial simulation environment that we have called RoVEn . RoVEn allows us to model robots, to implement robot controllers and to evolve them using genetic algorithms. In what follows we describe:

a) The RoVEn (Robot Virtual Environment) simulation environment.
b) A number of specific robots we have used in our experiments.
c) Our "evolutionary laboratory".

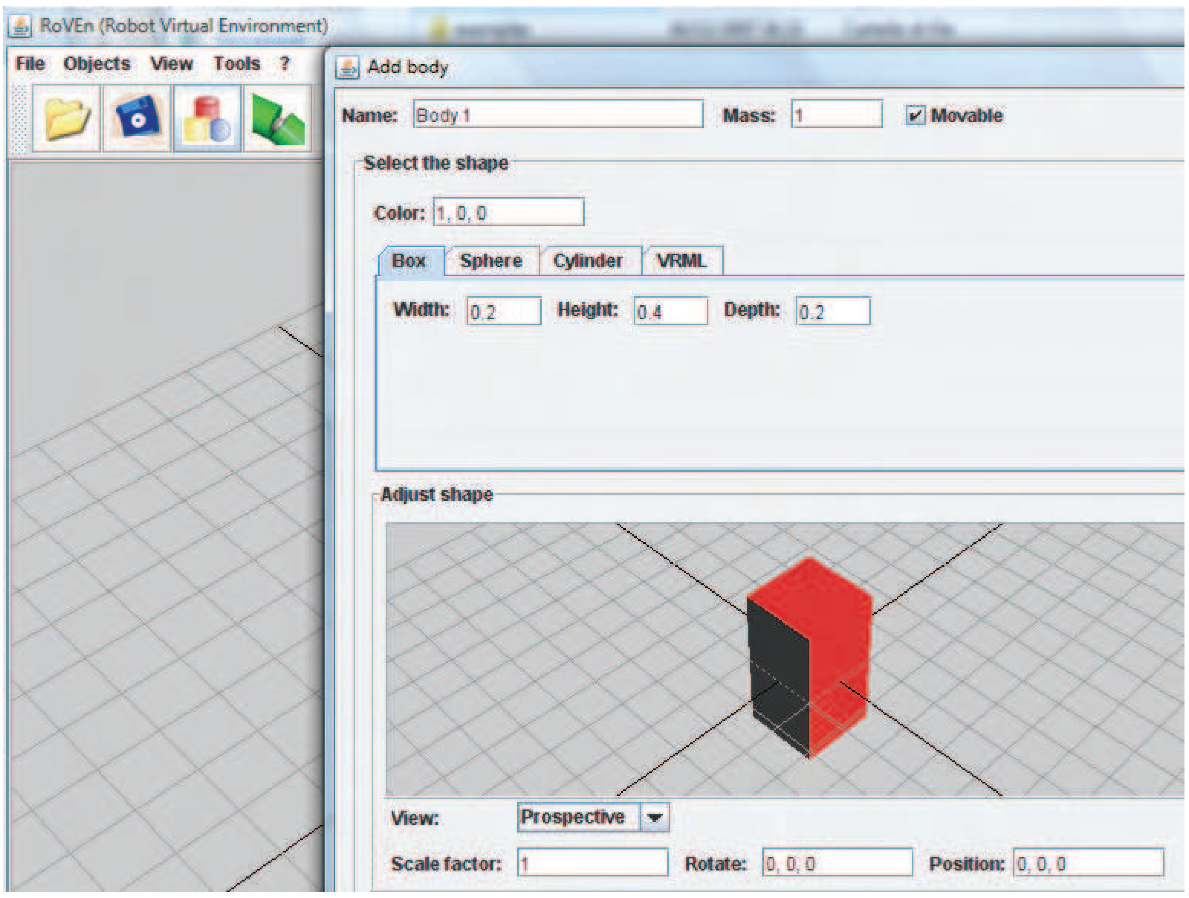
Any experiment in evolutionary robotics can be divided into the following steps:

1. Design and implement the robot body.
2. Insert the robot controller.
3. Associate the controller with a genome, susceptible to modification by genetic operators.
4. Analyze the evolutionary process.
5. Analyze the behavior of the "best behaving" robots using a simulation environment.

RoVEn makes it possible to perform all these steps in a single simulation environment. The environment was developed using Java 3D libraries for rendering and ODE (Open Dynamics Engine) libraries for the simulation of the physical world. The ODE simulation engine contains a numeric integrator that solves the equations of motion for inelastic bodies. Below we provide more details of the simulation environment.

4.1 Creating a robot body in the ROVEN simulation environment

The robot body is modeled as a set of inelastic bodies connected via joints. The use of joints makes it possible to define hierarchies of bodies and to create complex prototypes. Each body has a Shape**,** characterized by a color and geometry.



Name: Body 1 Mass: 1 [✓ Movable]

Color: 1, 0, 0

Width: 0.2 Height: 0.4 Depth: 0.2

View: Prospective

Scale factor: 1 Rotate: 0, 0, 0 Position: 0, 0, 0

Figure 7. The RoVEn environment used to create a single inelastic body. The lower part of the window contains controls making it possible to move, rotate and change the scale of bodies

RoVEn uses three elementary geometries (see figure 7):

a) Parallelograms.

b) Spheres.

c) Cylinders.

Obviously this is not enough to create complex shapes such as arms and legs. For this purpose it is possible to import additional shapes from VRML files. A Movable option makes it possible to distinguish between mobile objects (e.g. robots and robot parts) and immobile objects (a wall, an inclined plane etc.).

To join one body to another, the user selects the first body, clicks on the Join button on the Tool Bar and clicks on the second body. Any body which is jointed to another body (via a spherical joint) contains a motor, positioned on the pin on which the joint rotates. This is why it is not possible to have more than one motor per body. It is possible to define limits on how far the body can rotate along its three axes of rotation. Alternatively constraints can be inserted one at a time by setting the properties: Elevation bounds, Azimuth bounds, and Tilt bounds.

Figure 8 shows a six-legged agent created with RoVEn. It is this agent we used in the examples we refer to later. Figure 9 shows how we can use the same components to construct a humanoid or a four legged robot in the RoVEn environment.

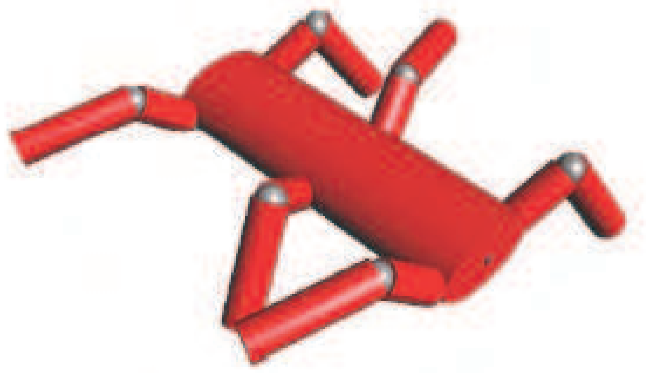


Figure 8. A six-legged body composed of a central body and 12 spherical joints

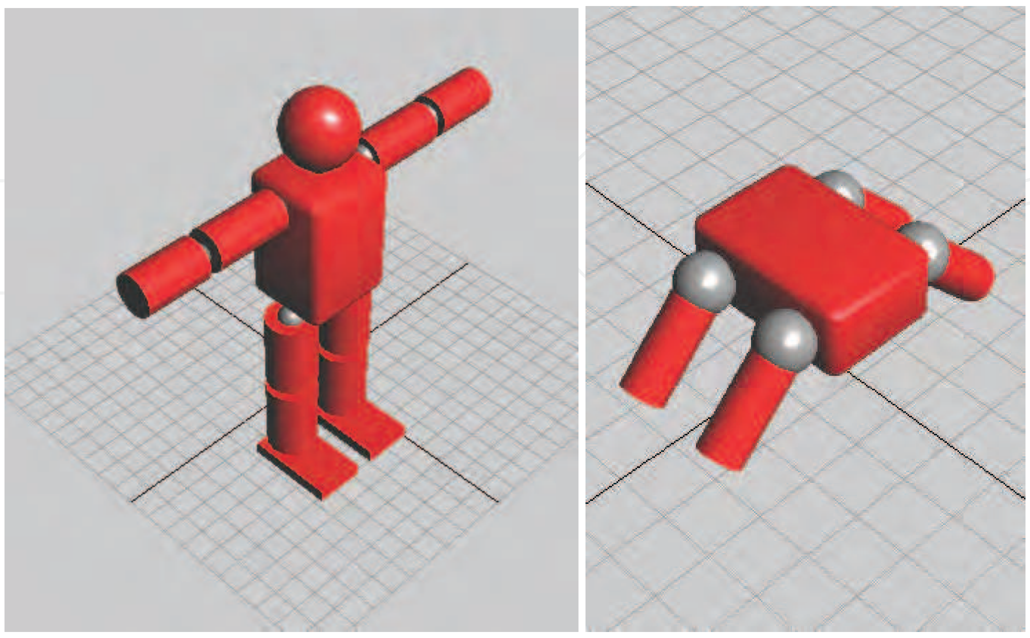


Figure 9. A four-legged and a humanoid robot in RoVEn

Actuators and sensors have to be installed on the bodies. To install a sensor, it is enough to select the body and click on *Add Sensor,* on the Tool Bar**.** This opens a window which can be used to install a new sensor. RoVEn provides four classes of sensor: a clock sensor which cycles between an output of 1 and an output of 0 at predefined times, a position sensor, a proximity sensor and a WebCam.

4.2 Controllers

RoVEn makes it possible to simulate different classes of controller and their interactions. *Controllers* read data from sensors and act on the actuators in such a way that the robot takes a specific action. Interacting controllers can read and write to special *registers* that simulate the registers normally provided by computer hardware. We have implemented several different classes of controller. Below we describe them briefly.

Interface

This class of controller provides an interface between devices and registers (which can be read and modified by other controllers). For each class of device, RoVEn displays data describing components suitable for connection to registers. For instance, for motors installed on joints, the system displays three text boxes, showing the names of the registers where the system stores the robot's speed along the three axes of rotation and an additional three text boxes showing the registers with the angles reached by the motor. Alternatively the values can be inserted by hand. The proper location for interface controllers connected to sensors is a low layer of the controller which uses the interfacing registers. In this way, the controller can use up to date information for every interaction. Otherwise the values in the register would always refer to the previous interaction.

User control

This controller is used exclusively to control motors. When a simulation begins, the user uses a special window to defines the speed of the motor along the three axes of rotations.

Time series

This controller allows the user to define how the speed of the motor changes over time. One method is to import a text file defining the speed of the motor along the three axes of rotation at specific times. Alternatively the user can set a mathematical function defining the speed of the motor as a function of time.

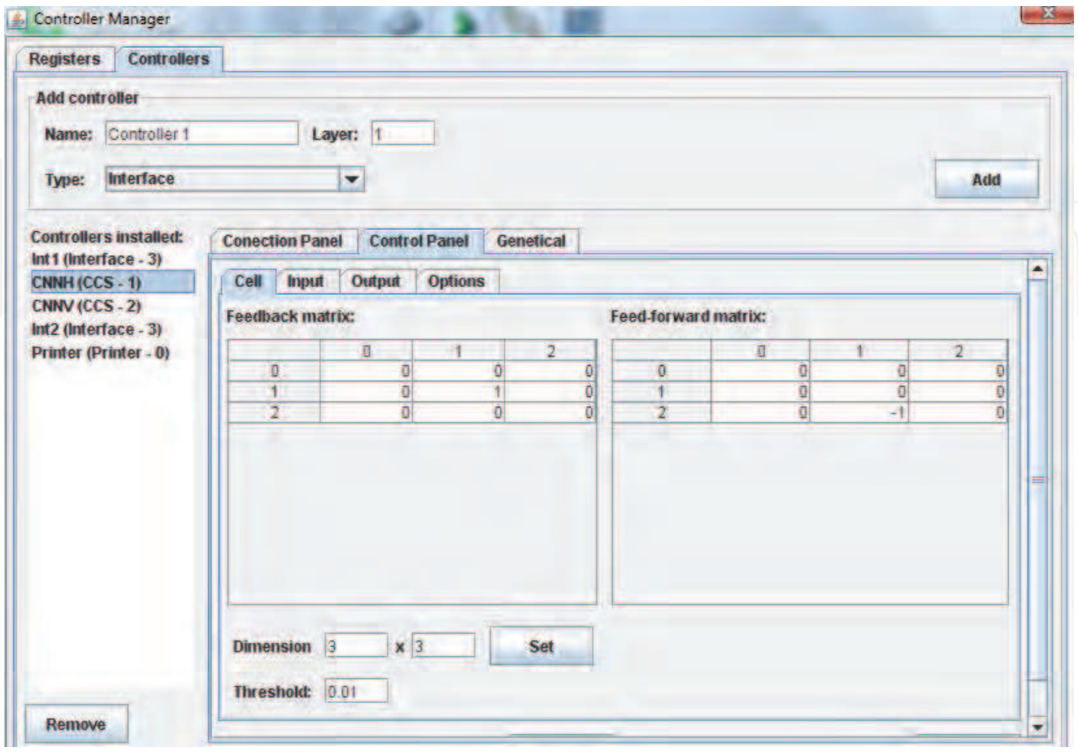
Chua circuit

This controller makes it possible to define a set of Chua circuits connected by resistors and to use the system. The current and voltage produced by this system controls a motor. A special control panel allows users to define the properties of the individual components.

CNN

These controls make it possible to emulate a CNN and to connect the CNN to the input and output registers, available to other controllers. The control panel contains a number of different tabs (see Figure 10). The *Cell* tab allows the user to define the Feedback and Feed-forward matrices**,** the threshold for the cell and the size of the network. The *Input* and *Output* tabs allow the user to define the registers where the network has to read its input and write its output. The *Option* tab allows the user to set the parameters for the CNN, including the integration step (in the *Delta* field) and the boundary conditions (see Figure 10).

The controller is designed to support the applications described in Section 3. As we will see in the following section, the values of the Feedback e Feed-forward matrices can be optimized using evolutionary techniques (genetic algorithms). The network can take input from several different kinds of sensor. In the experiment described in the following section, we use a position sensor to evaluate the trajectory followed by the robot.



Name: Controller 1
Layer: 1
Type: Interface

Controllers installed:
Int1 (Interface - 3)
CNNH (CCS - 1)
CNNV (CCS - 2)
Int2 (Interface - 3)
Printer (Printer - 0)

Feedback matrix:

	0	1	2
0	0	0	0
1	0	1	0
2	0	0	0

Feed-forward matrix:

	0	1	2
0	0	0	0
1	0	0	0
2	0	-1	0

Dimension 3 x 3
Threshold: 0.01

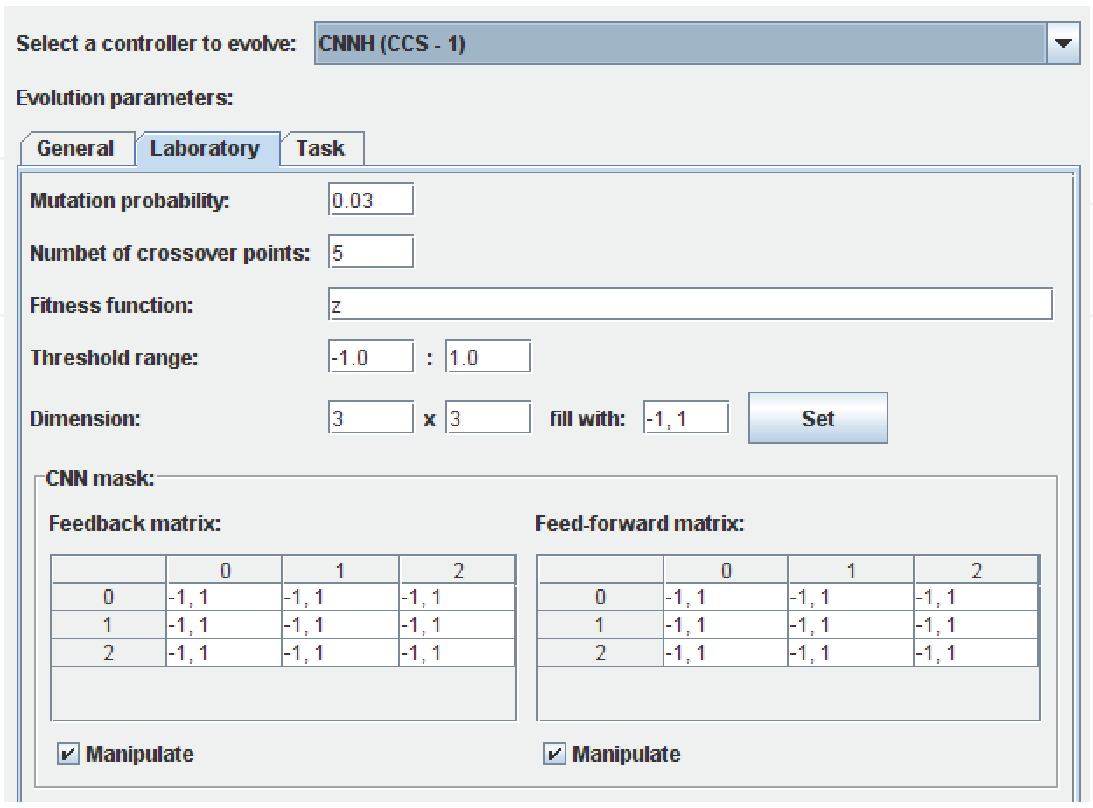
Figure 10. The CNN Control Panel

<u>Neural Networks</u>
In addition to the controllers mentioned earlier, the system also includes a neural network controller. This makes it possible to replicate classical experiments in evolutionary robotics. For further details see "Evolutionary Robotics" (Nolfi & Floreano, 2000).
<u>Other controllers</u>
In cases in which existing controllers are not sufficient, RoVEn allows users to develop their own *ad hoc* controllers. Ad hoc controllers are developed in **Java** and loaded into RoVEn as plug-ins. RoVEn provides developers with a special SDK to help them in building plug-ins and in importing them into applications.

4.3 A genetics laboratory for CNN

Genetic Algorithms (GAs), invented by John Holland (Holland, 1993), are an optimization technique which has been applied to a large class of problems where classical techniques do not provide appropriate solutions. GAs are inspired by Darwin's theory of evolution through natural selection. The general idea is to evolve problem solutions using technique s similar to those nature uses to evolve animal species.
Given a system that needs to perform a specific operation, GAs optimize the parameters of the system by evolving optimally performing individuals. Having specified the problem in this way, we can see GAs as a means to generate a desirable behavior. A necessary condition is that the system can be fully specified by a finite set of parameters. The parameters are then represented as a string of characters (the system's "DNA" or genotype). In this setting, each parameter is a gene; the working system in which these genes are expressed is the "phenotype". In the work we describe below the phenotype consists of a robot controlled

by a CNN; the genotype is the set of parameters controlling the CNN, that is the values of the feedback and feedforward matrices.


Select a controller to evolve: CNNH (CCS - 1)

Evolution parameters:

General | Laboratory | Task

Mutation probability: 0.03

Numbet of crossover points: 5

Fitness function: z

Threshold range: -1.0 : 1.0

Dimension: 3 x 3 fill with: -1, 1 Set

CNN mask:

Feedback matrix:

	0	1	2
0	-1, 1	-1, 1	-1, 1
1	-1, 1	-1, 1	-1, 1
2	-1, 1	-1, 1	-1, 1

[✓] Manipulate

Feed-forward matrix:

	0	1	2
0	-1, 1	-1, 1	-1, 1
1	-1, 1	-1, 1	-1, 1
2	-1, 1	-1, 1	-1, 1

[✓] Manipulate

Figure 11. The window used to control genetic operators and initial values for the CNN controller

An important phase in the GA is the evaluation of the fitness of different phenotypes. In the work described here, we simulate the behavior of the robot in a simulator, extract behavioral indicators (e.g. Lagrangian parameter values and their derivatives, distance covered etc.) and use the values of these indicators as the arguments of a *fitness function f* that computes the fitness of the robot.

A Genetic Algorithm comprises the following steps:

1. Generation: generation of an initial population of individuals with randomly generated CNN parameters;
2. Fitness evaluation: simulation of the behavior of each individual robot and computation of fitness values for each individual;
3. Selection: selection of the individuals with the highest fitness value (a certain percentage of the population);
4. Reproduction: generation of new individuals by *crossing-over* the genotypes from two parent individuals and randomly *mutating* a certain percentage of their genes;
5. Go to step 2: repetition of the process through the creation of a new generation of individuals.

In defining a GA, one of the key steps is the choice of the fitness function. Different problems require different fitness functions adapted to their specific requirements. Another important issue is the optimization of the code so as to allow as many iterations of the algorithm as possible in the shortest possible time (Mitchell, 1996).

The "Evolutionary Laboratory" we used to evolve the CNN controller allowed us to define the following parameters:

- Mutation probability: the probability (a value between 0 and 1) that any given gene will undergo mutation in a single iteration of the algorithm.
- Number of cross-over points: the number of cross-over points.
- Evaluate fitness on axis: the axis along which to measure the distance traveled by the robot. The result of this measurement is used to evaluate the robot's fitness.
- Threshold range: the range of possible values for the Threshold parameter.

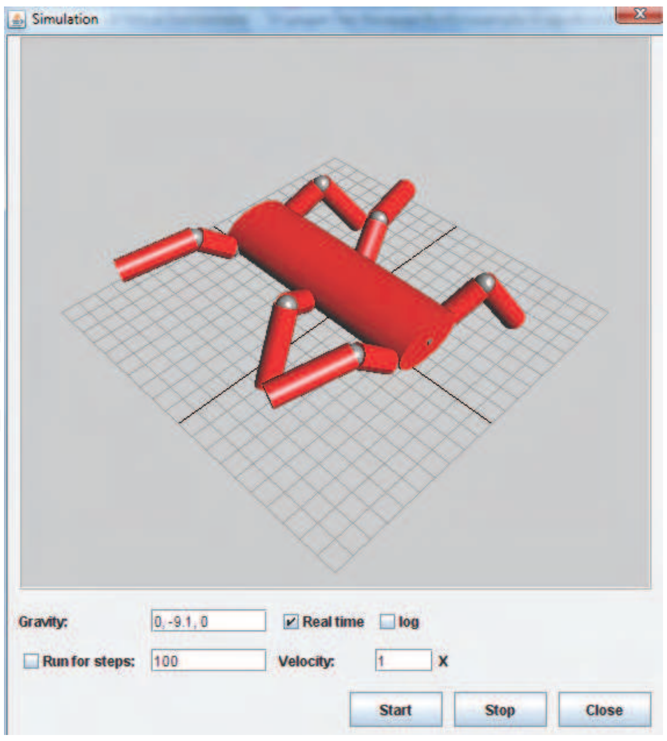
Figure 11 shows one of the user interfaces used during the evolution of the CNN controller for the robot

4.4 Analysis of evolutionary trends

The output of the GA consists of the genotypes present in the last generation of individuals produced by the algorithm. The file containing these values can be loaded using the *Genetical Tab* in the *Processor Manager* window. The *Load* button allows the user to load the files for all individuals with fitness higher than a user-specified threshold. The user can then select an individual and use the *Set* button to copy its DNA to the current controller.
An additional Data Analysis module allows the user to display a graph showing mean and maximum fitness values at each step in the evolutionary process.

4.5 Simulating the behavior of optimal individuals

The final environment allows the user to observe the behavior of optimal individuals. (see Figure 12). A printer interface makes it possible to print the results of the simulation to a file where they can then be analyzed using spreadsheet software.


Gravity: 0, -9.1, 0 [✓ Real time] log

Run for steps: 100 Velocity: 1 X

Figure 12. The behavior simulation window

5. Experimental Setting

We performed a number of simulations, using a range of different controllers. In what follows we present an example from this work.

5.1 Robot design

The robot (see figure 13) consists of a body, connected to six legs. Each leg is made up of two parts. Where parts are connected, the connection is a spherical joint. In the case of legs, one "internal joint" joins the leg to the body; a second "external joint" joins the second part of the leg to the first.

In what follows we will consider a configuration in which the internal joint is restricted to movement along the azimuth. In this way the internal joint can only move ahead of and behind the legs. We also eliminate two degrees of freedom for the external joint, restricting it to elevation rotation (raising and lowering the second part of the legs). In this way, if we know the position of the robot's center of gravity on the plane, and the direction in which it is moving, the system has just 12 degrees of freedom (the two angles of rotation associated with each of its six legs). (If we assume that the body is always in contact with the ground it has 15 degrees of freedom). The movement of the robot's legs, starting with this initial configuration, uniquely determines the robot's position. Elevations are constrained within the range 0° to 45°; the azimuth is constrained to the range 20° to +20°. This prevents the legs from touching and creating problems for the simulator.

For the robot controller, we used two CNNs connected to the internal joints, and one connected to the external joints. Since the robot has six legs, there are 6 + 6 hinges, each with a motor connected to a cell in the CNN. The output from each cell provides the input for the motors, as shown in Figure 13.

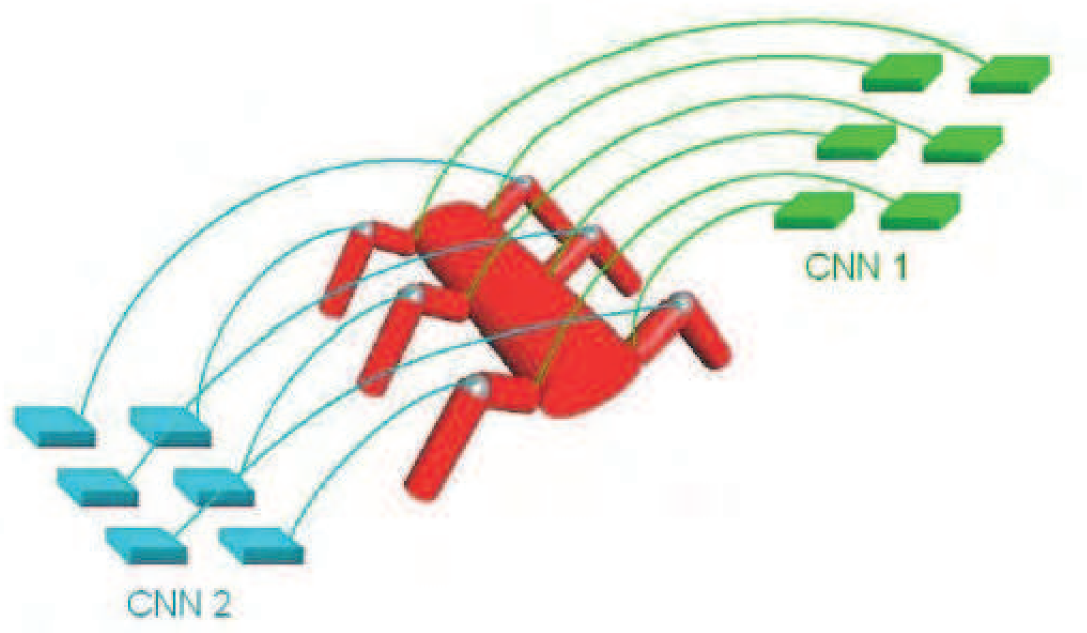


Figure 13. The robot controller. The output from 2 6-cell CNNs is directly connected to the motors

Each CNN consists of 6 cells, each cell lies in a neighborhood with r=1. Thus each cell has 9 neighbors (including itself). Given that all cells have a neighborhood of the same size, the boundary conditions for the network will be periodical (see Figure 14).
The output from each cell defines the input to the motors as shown in Table 1

CNN1		CNN2	
IFLJ	IFRJ	EFLJ	EFRJ
IMLJ	IMRJ	EMLJ	EMRJ
IBLJ	IBRJ	EBLJ	EBRJ

Table 1. Connections between the outputs of the two CNNs and the 12 motors

To identify the motors we use a code based on the following conventions. The first letter in the code indicates whether the joint is internal (I) or external (E); the second indicates whether the joint is on a front, middle or back leg (F, M, B). The third letter shows whether the joint is on the left (L) or the right (R). The last letter (J) indicates that the code refers to a joint. Each cell in the first CNN takes its input from the angular sensors inside the hinge on the joint to which it delivers its output. The second CNN takes its input from the first (the first cell of CNN 1 is connected to the first cell of CNN 2, the second cell of CNN 1 to the second cell of CNN 2 etc.).

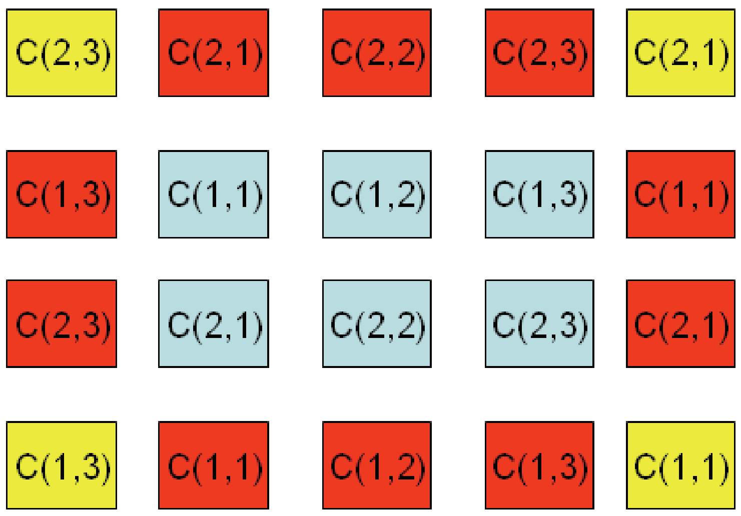


Figure 14. We assume that each CNN has periodical boundary conditions and that cells on the boundaries are connected to other cells in their neighborhood as shown in the Figure.
For example the neighborhood $S_{11}(1)$ for cell C_{11} is given by:

$$S_{11}(1)=\{C_{23},C_{21},C_{22},C_{13},C_{11},C_{12},C_{23},C_{21},C_{22}\}$$

5.2 The Genetic Algorithm

We conducted a number of simulations in which we evolved both the first and the second CNN. At the current stage in our work, we have no way of co-evolving the controllers. We obtained the best results when we began by evolving the first CNN and used the results from the best individual to evolve the second CNN.
We used a population of 30 individuals. On every step in the evolutionary process, the population included elite of 10 individuals which did not evolve during that step, 10

individuals generated from the elite by mutation and cross-over and 10 new, randomly generated individuals. Each simulation lasted 20 seconds. The probability of mutation was 3%. The number of cross-over points was set to 3.

5.3 Results

Multiple simulations produced fairly homogeneous results. The GA was reasonably effective in producing a robot with the ability to move rapidly away from its initial position. Evolving of CNN2 for 80 steps, we observed a rapid increase in fitness for the first 15 steps, up to a maximum fitness of approximately 3.5 (and a mean fitness for the elite of 3). Figure 15 shows the fitness achieved by the best individual and the mean fitness for the elite.

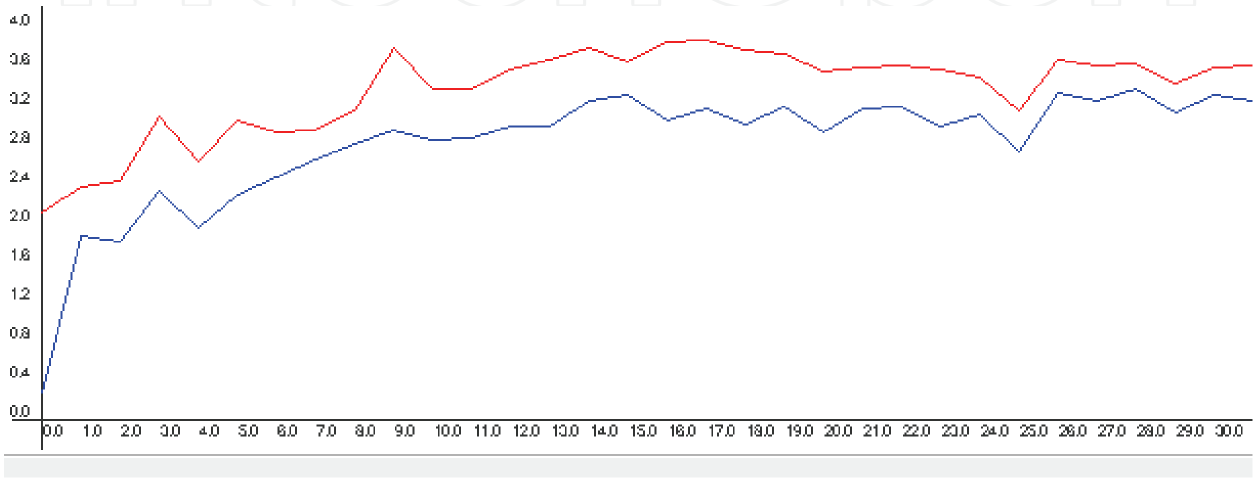


Figure 15. Fitness for the best individual and the mean fitness of the elite

Fig. 15 shows the fitness of the best individual in each generation. As can be seen from the graph, the highest fitness achieved by any individual in any generation was 3.420.

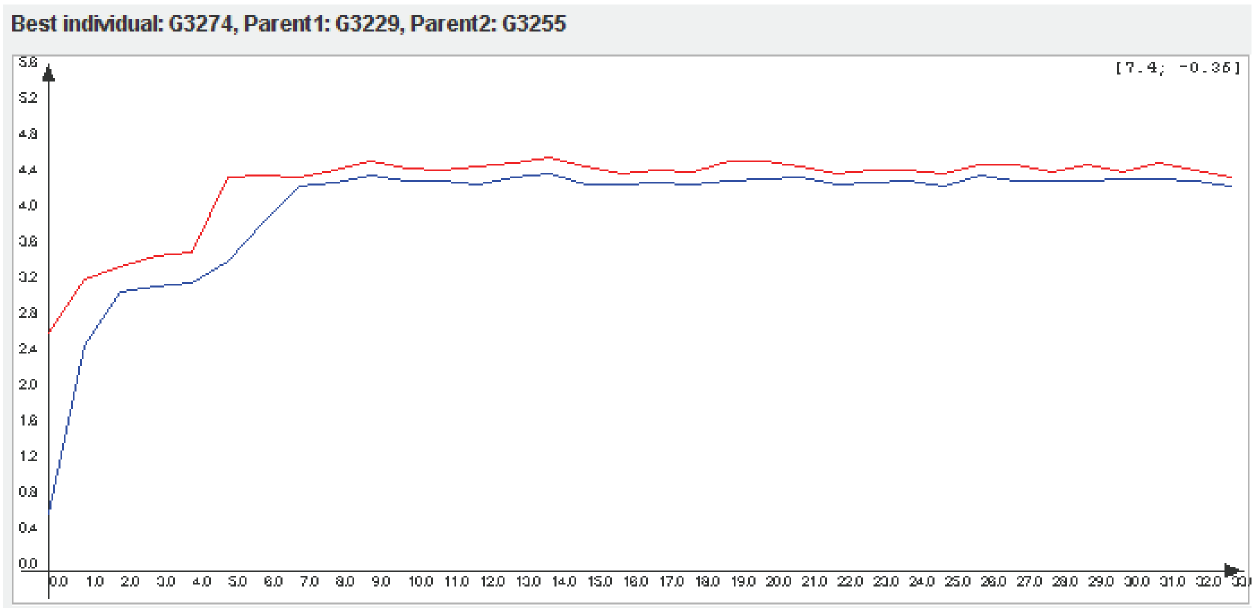


Figure 16. Evolution of CNN1

Following this initial step we took the best individuals from the last generation and continued the evolutionary process for another 30 steps, this time by evolving CNN 1. The results are shown in Figure 18. After about 7 steps the fittest individual reached a peak of fitness which remained unchanged for the rest of the simulation; after roughly the same number of steps the mean fitness of the elite also reached a ceiling. The maximum fitness achieved was approximately 4.5. The mean fitness of the elite was close to the fitness of the best individual (see Figure 16).

6. On chip prototypes

The CNN Universal Machine (CNN-UM) architecture, introduced by Roska & Chua in 1993 (Roska & Chua, 1993), is an extension of the original CNN paradigm (Chua & Yang., 1988) that has proved extremely effective in many image processing applications. Many different implementations have been proposed. One of the most efficient uses an analog VLSI. With state of the art miniturization it has proved possible to implement a 128*128 or even a 1000*1000 CNN array on a 1 or 2 cm strip of silicon (Rodrigez-Vazquez et al, 1998; Ioan et al., 2001; Paasio et al, 1997). These chips are extremely fast, supporting the equivalent of 1012 billion (1 tera) floating point equivalent operations per second – equivalent to the computational capacity of a supercomputer with 9200 Pentium chips, at the time of writing the fastest in the world (Intel, 2007).
Optical implementations (Andersson, 1998) provide large image resolutions, and make it possible to use larger templates. Feed forward templates compute at the speed of light, providing very fast computation. With feedback templates, by contrast, the optical feedback signal has to be amplified electronically, slowing down the system. Compared to purely electronic systems, these optical systems are relatively bulky and fragile. Systems that use digital hardware emulate CNN-UMs (Wen et al., 1994; Ikenaga & Ogura, 1996; Doan et al., 1994; Adaptive Solutions Inc, 2007; Zarandy et al., 1998) while slower than their analog counterparts, are also more versatile. Given that they use standard digital CMOS technology they are also much quicker to design. One example is the CASTLE architecture (Zarandy et al., 1998) which solves complex image processing problems for medium resolution video streams. Assuming a 25fps digital video feed, CASTLE is fast enough to perform 500 CNN iterations (3x3 convolutions) on frames with 12-bit precision) on each 240x320frame.

7. Conclusions

In this chapter, we have shown how dynamical systems can be used to explore the layering of the sensorial and perceptual activity underlying intelligent behavior in simulated and physical robots. We have described RoVEn, an open, integrated simulation environment which can be easily extended with additional functionality. We have shown some of the results that can be achieved by using Genetic Algorithms to evolve controllers for robot controllers. This represents a first direction in the direction we are seeking to follow. The initial results appear to be satisfactory.
Our implementation work represents an extension of previous work in bio-inspired robotics. As a next step, we intend to build artificial organisms endowed with a cognitive architecture, and a rich sensorial system allowing the agent to recognize and discriminate between specific emotional stimuli from the environment, other agents and humans. These robots will have the ability to generate a set of different motor behaviors, in both simulated and physical environments. The architecture will be multiple-layered, based on modules of interconnected CNN devices. As a result, the robots' cognitive system will be dynamical,

adaptive and self-organizing. The robot sensor system will acquire information from the environment thereby helping to provide the robot with vision, audition, recognition and consciousness. Data from the robot sensory system will be processed in specialized processing units, equivalent to brain areas. It will also be sent to cognitive perceptual centers where it will activate different kinds of behavior. This will make it possible to implement a motor-perception cycle, in which emotional- motivational modules select actions in response to species-specific stimuli from the environment. The system will also include a communication module used to interact with other peers or with humans. The strategic goal is to create new generation of emotional, communicating robots with high-level cognitive capabilities that enable them to achieve complex goals in complex environments, using limited computational resources.
In this scenario, it will be possible to create *artificial brains* with cellular architectures related to a set of basic functionalities which define a biological agent - with evolving or co-evolving modules - which allow for the simulation of the growth and the adaptation of the robot to the environment. These architectures will share a set of common properties such as topographic cellular morphism, different dynamics of communication among cells or layers of the structures and many working non-linear dynamics. Furthermore, it will be possible to combine continuous and discrete robot's representation, integrating algorithmic and physical action in space and time. In this way, the CNN paradigm will make it possible to achieve a vast increase in robot processing power and in the number of cognitive processes that can run in parallel.

8. References

Adaptive Solutions Inc (2007). CNAPS/PCI Parallel Co-processor

Andersson, S. (1998). Recent Progress on Logic and Algorithms for Optical Neural Networks (ONN), *Proc. of the fifth IEEE Int. Workshop on Cellular Neural Networks and their Applications*, London.

Arena, P.; Branciforte, M. & Fortuna, L. (1998). A CNN based experimental frame for patterns and autowaves, *International Journal on Circuit Theory and Applications*, n°3, pp. 120-136.

Arena, P. & Fortuna, L. (2002). Analog Cellular Locomotion Control of Hexapod Robots, *IEEE Control Systems Magazine*, pp. 21-36

Bilotta, E.; Cutrì, G. & Pantano, P. (2006). Evolving robot's behavior by using CNNs, *Proc. The Ninth International Conference on the SIMULATION OF ADAPTIVE BEHAVIOR (SAB'06) - FROM ANIMALS TO ANIMATS* 9, 25 - 29 September 2006, CNR, Roma, Italy, *Lecture Notes in Artificial Intelligence*, pp. 631-639

Bilotta, E.; Pantano, P. & Stranges, F. (2007a). A gallery of Chua's Attractors - Part I, *International Journal of Bifurcation and Chaos*, vol. 17, n° 1, pp. 1-60

Bilotta, E.; Pantano, P. & Stranges, F. (2007b). A gallery of Chua's Attractors - Part II, *International Journal of Bifurcation and Chaos*, vol. 17, n° 2, pp. 293-380

Bilotta, E.; Pantano, P. & Stranges, F. (2007c). A gallery of Chua's Attractors - Part III, *International Journal of Bifurcation and Chaos*, vol. 17, n° 3, 657-734

Bilotta, E.; Di Blasi, G.; Pantano, P. & Stranges, F. (2007d). A gallery of Chua's Attractors – Part IV, *International Journal of Bifurcation and Chaos*, vol. 17, n° 4, pp. 1017-1078

Bilotta, E.; Di Blasi, G.; Pantano, P. & Stranges, F. (2007e). A gallery of Chua's Attractors – Part V, *International Journal of Bifurcation and Chaos*, vol. 17, n° 5, pp. 1383-1511

Bilotta, E.; Di Blasi, G.; Pantano, P. & Stranges, F. (2007f). A gallery of Chua's Attractors – Part VI, *International Journal of Bifurcation and Chaos*, vol. 17, n° 1, pp. 1801-1910

Calabrese R.L., (1995). Oscillation in motor pattern generating networks, *Current Opinion in Neurobiology*, vol. 5, pp. 816, 823

Cangelosi, A. & Parisi, D. (2002). Computer simulation: A new scientific approach to the study of language evolution. In *Simulating the evolution of language*, Cangelosi, A. & Parisi, D. (Ed.), London: Springer, pp. 3-28

Chua, L.O.& Yang, L. (1988). Cellular neural networks: Theory and Applications, *IEEE Trans. on Circuits and Systems*,vol. 35, pp. 1257-1290

Chua, L.O. (1998). *CNN: A Paradigm for Complexity*. World Scientific Publishing Co. Pte. Ltd.

Chua, L.O. (1995). Special issue on nonlinear waves, patterns and spatio-temporal chaos in dynamic arrays , *IEEE Transactions on Circuits and System*, vol. 42, n°10

Doan, M.D.; Glesner, M.; Chakrabaty, R.; Heidenreich, M.& Cheung, S. (1994). Realisation of digital Cellular Neural Network for image processing, *Proc. of the IEEE CNNA'94*, Rome, pp. 85-90

Fried, S. I.; Münch, T. A. & Werblin, F., S. (2005). Directional Selwectivity is Formed at Multiple Levels by Laterally Offset Inhibition in the rabbit Retina, *Neuron*, vol. 46, pp. 117-127

Gacsádi A.; Tiponut V. & Szolgay P. (2006). Image-Based Visual Servo Control of a Robotic Arm by Using Cellular Neural Networks, *Proceedings of the 15th International Workshop on Robotics in Alpe-Adria-Danube Region*, (RAAD 2006), ISBN 9637154 48 5, CDRom, Balatonfüred, Hungary

Gacsádi A. & Szolgay P. (2004). A variational method for image denoising, by using cellular neural networks, *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications* (CNNA 2004), Budapest, Hungary, ISBN 963-311-357-1, pp. 213-218

Harnad, S. (1990). The Symbol Grounding Problem, *Physica D*, vol. 42, pp. 335-346

Holland, J. (1993). *Adaptation in natural and artificial systems*, Penguin Books

Ikenaga, T. & Ogura, T. (1996). Discrete-time Cellular Neural Networks using highly-parallel 2D Cellular Automata CAM2, *Proc. of Int. Symp. on Nonlinear Theory and its Applications*, pp. 221-224.

Intel (2007) http://www.intel.com/pressroom/archive/releases/CN0611B.HTM

Ioan, D.; Duca, A. & Rebican, M. (2001). Teams of Autonomous Softwarw Agents (TASA) to Solve Inverse ENDE Problems. *The Abstracts of Progress In Electromagnetics Research Symposium* (PIERS01), Osaka, Japan, iul. pp. 18-22.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. The MIT Press

Keijzer, F. (2002). Representation in dynamical and embodied cognition, *Cognitive Systems Research*, vol. 3, pp. 275–288

Krinsky, V.I. (1984). *Self-organization: autowaves and structures far from equilibrium*, Springer Ed, Berlin

Murray, J.D. (1989). *Mathematical biology*, Springer Ed. , Berlin

Nolfi, S. & Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA, MIT Press/Bradford Books.

Paasio, A.; Dawindzuk, A.; Halonen, K. & Porra, V. (1997). Minimum Size 0.5 Micron CMOS Programmable 48x48 CNN Test Chip, *European Conference on Circuit Theory and Design*, Budapest

Pearson, G.K. (1993). Common principles of motor control in vertebrates and invertebrates, *Annal Review of Neuroscience*, n° 16, pp. 295-297

Rodrigez-Vazquez, A.; Dominguez-Castro, R. & Espejo, S. (1998). Challenges in Mixed-Signal IC Design of CNN Chips in Submicron CMOS, *Proc. of the fifth IEEE Int. Workshop on Cellular Neural Networks and their Applications*, April,London.

Roska, T. & Chua, L.O. (1993). The CNN Universal Machine: an analogic array computer, *IEEE Transactions on Circuits and Systems-II*, vol. 40, pp. 163-173

Roska, B.; Molnara, A. & Werblin, F., S. (2006) Parallel Processing in Retinal Ganglion Cells: How Integration of Space-Time Patterns of Exicitation and Inhibition Form the Spiking Output, *Journal of Neurophysiology*, vol. 95, pp. 3810-3822

Schilling, M.; Cruse, H. & Arena, P. (2007). Hexapod Walking: an expansion to Walknet dealing with leg amputations and force oscillations, *Biol. Cybern*. vol. 96, pp. 323-340

van Gelder, T. J. (1995). The distinction between mind and cognition. In Houng Y. H. and Ho J. C.(Ed), *Mind and Cognition*. Taipei, Academia Sinica, pp. 57-82.

van Gelder, T. J. (1998a). The dynamical hypothesis in cognitive science, *Behavioural and Brain Sciences*, vol. 21, pp. 1-14

van Gelder, T. J. (1998b). Disentangling dynamics, computation, and cognition, *Behavioural and Brain Sciences*, vol. 21, pp. 40-47.

Wen, K.A.; Su, J.Y. & Lu, C.Y. (1994). VLSI design of digital Cellular Neural Networks for image processing, *Journal of Visual Communication and Image Representation*, vol.5 , n°2, pp. 1117-126

Zarandy, A.; Keresztes, P.; Roska, T. & Szolgay, P. (1998). An emulated digital architecture implementing the CNN Universal Machine, *Proc. of the fifth IEEE Int. Workshop on Cellular Neural Networks and their Applications*, London, pp. 249-252