HOW TO EVOLVE AUTONOMOUS ROBOTS: DIFFERENT APPROACHES IN EVOLUTIONARY ROBOTICS

[*]Stefano Nolfi [**]Dario Floreano [***]Orazio Miglino [****]Francesco Mondada

[*]Institute of Psychology, National Research Council
15, Viale Marx - 00187 - Rome - Italy
e-mail: stefano@kant.irmkant.rm.cnr.it
[**]Laboratory of Cognitive Technology,
AREA Science Park - Trieste, Italy
e-mail: dario@psicosun.univ.trieste.it
[***]Department of Psychology, University of Palermo
Viale delle Scienze, Palermo, Italy
e-mail: orazio@caio.irmkant.rm.cnr.it
[****]Laboratory of Microcomputing
Swiss Federal Institute of Technology, Lausanne, Switzerland
e-mail: mondada@di.epfl.ch

May 1994

Technical Report PCIA-94-03

To appear In: Proceedings on the Artificial Life IV Conference, July 6-8, Cambridge, MA, U.S.A.

HOW TO EVOLVE AUTONOMOUS ROBOTS: DIFFERENT APPROACHES IN EVOLUTIONARY ROBOTICS

[*]Stefano Nolfi [**]Dario Floreano [***]Orazio Miglino [****]Francesco Mondada

[*]Institute of Psychology, National Research Council
15, Viale Marx - 00187 - Rome - Italy
e-mail: stefano@kant.irmkant.rm.cnr.it
[**]Laboratory of Cognitive Technology,
AREA Science Park - Trieste, Italy
e-mail: dario@psicosun.univ.trieste.it
[***]Department of Psychology, University of Palermo
Viale delle Scienze, Palermo, Italy
e-mail: orazio@caio.irmkant.rm.cnr.it
[****]Laboratory of Microcomputing
Swiss Federal Institute of Technology, Lausanne, Switzerland
e-mail: mondada@di.epfl.ch

Abstract

A methodology for evolving the control systems of autonomous robots has not yet been well established. In this paper we will show different examples of applications of evolutionary robotics to real robots by describing three different approaches to develop neural controllers for mobile robots. In all the experiments described real robots are involved and are indeed the ultimate means of evaluating the success and the results of the procedures employed. Each approach will be compared with the others and the relative advantages and drawbacks will be discussed. Last, but not least, we will try to tackle a few important issues related to the design of the hardware and of the evolutionary conditions in which the control system of the autonomous agent should evolve.

1. Introduction

In the last few years new approaches that involve a form of simulated evolution have been proposed in order to build autonomous robots that can perform useful tasks in unstructured environments (Brooks, 1992; Cliff, Husband and Harvey, 1993). The great amount of interest in this new approach is due to dissatisfactions with traditional robotics and Artificial Intelligence and to the belief that interesting robots may be too difficult to design. There are two main reasons why strong difficulties arise in designing a control system for autonomous robots:

(a) it is extremely difficult to co-ordinate the parts of a robot, both at the level of mechanics and of the control system; it is also hard to predict the interaction between these two levels. As Cliff, Harvey, and Husband noted (1993) the complexity of the design scales faster than the number of parts or modules within the system; rather, it scales with the number of possible interactions between parts and modules.

(b) autonomous robots interact with an external environment and, therefore, the way in which they behave in the environment determines the stimuli they will receive in input (Parisi, Cecconi, and Nolfi, 1990). Each motor action has two different effects: (1) it determines how well the system performs with respect to the given task; (2) it determines the next input stimuli which will be perceived by the system (this last point strongly affects the success or the failure of a sequence of actions). Determining the correct motor action that the system should perform in order to experience good input stimuli, is thus extremely difficult because any motor action may have long term consequences. Also, the choice of a given motor action is often the result of the previous sequence of actions. A final source of uncertainty in the design of the system is the fact that often the interaction between the system and the environment is not perfectly known in advance.

Thus, it would appear reasonable to use an automatic procedure, such as a genetic algorithm, that gradually builds up the control system of an autonomous agent by exploiting the variations in the interactions between the environment and the agent itself. It remains to be determined if it is feasible. In particular we should answer the questions: What to evolve? And, how to evolve it?

The choice of what to evolve is controversial. Some authors have proposed to evolve controllers in the form of explicit programs in some high-level language. Brooks (1992) proposes to use an extension of Koza's genetic programming technique (Koza, 1990). Dorigo and Schnepf (1993) propose to use a form of classifier system. Others propose to evolve neural networks controllers (Cliff, Husband and Harvey, 1993; Floreano and Mondada, in press; Miglino, Nafasi, Taylor, 1994; Nolfi, Miglino and

Parisi, 1994). We think that evolving neural networks is the most promising way for a number of reasons:

(a) Neural networks can easily exploit various form of learning during life-time and this learning process may help and speed up the evolutionary process (Ackley and Litmann, 1991; Parisi and Nolfi; in press).

(b) Neural networks are resistant to noise that is massively present in robot/environment interactions. This fact also implies that the fitness landscape of neural networks is not very rugged because sharp changes of the network parameters do not normally imply big changes in the fitness level. On the contrary it has been shown that introducing noise in neural networks can have a beneficial effect on the course of the evolutionary process (Miglino, Pedone, and Parisi; 1993).

(c) We agree with Cliff, Harvey, and Husband (1993) that the primitives components manipulated by the evolutionary process should be at the lowest level possible in order to avoid undesiderable choices made by the human designer. Synaptic weights and nodes are low level primitive components.

The methodology used to evolve control systems for autonomous robots is not well established. The large population size and the number of generations required for the emerging of interesting form of behaviors with the evolutionary techniques implies that a large number of robots must be evaluated. This fact has often restricted most of the experiments to computer simulations and declaration of the intentions to move to physical robots. However, traditional wisdom tells us that computer simulations are of limited usefulness for predicting the behavior of real robots.

In this paper we want to show different examples of applications of evolutionary robotics to real robots by describing three different approaches to develop neural controllers for mobile robots. In all the experiments described below real robots are involved and are indeed the ultimate means of evaluating the success and the results of the procedures employed. Each approach will be compared with the others and the relative advantages and drawbacks will be discussed. Last, but not least, we will try to tackle a few important issues related to the hardware design of autonomous agents and to the new methodological issues in the analysis of the system.

2. The evolution of an ability to navigate by using the physical robot

Floreano and Mondada (in press) developed neural controllers for autonomus agents that should perform a navigation task by using an evolutionary approach. The robot used was Khepera, a miniature mobile robot (Mondada, Franzi and Ienne, 1993). Khepera has a circular shape with a diameter of 55 mm, a height of 30 mm and a weight of 70g; it is supported by two wheels and two small teflon balls. The wheels are controlled by two DC motors with an incremental encoder (10 pulses per mm of advancement of the robot), and can move in both directions. The robot is provided with eight infra-red proximity sensors. Six sensors are positioned on the front of the robot, the remaining two on the back. A motorola 68331 controller with 256 Kbytes of RAM and 512 Kbytes ROM manages all the input-output routines and can communicate via a serial port with a host computer. Khepera was attached to the host by means of a lightweight aerial cable and specially designed rotating contacts. This configuration allowed a full track and record of all important variables by exploiting the storage capabilities of the host computer; at the same time it provides electrical power without using time-consuming homing algorithms or large heavy-duty batteries.

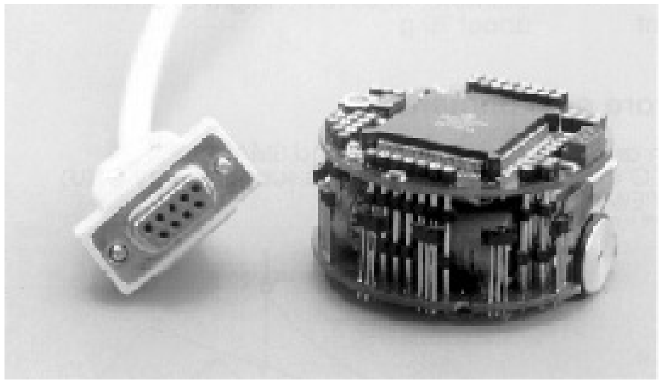

Figure 1. Khepera, the miniature mobile robot.

The robot was put in an environment consisting of an arena with internal walls of irregular shape. The external size of the arena was approx. 80x50 cm. The walls were made of light-blue polystyrene and the floor of a gray thick paper. Such environment was illuminated from above by a 60 watt light bulb.

The authors' goal was to develop a robot that could avoid obstacles while keeping the straightest possible trajectory at the fastest speed. The evolutionary training was a standard genetic algorithm as described by Goldberg (1989) with fitness scaling and roulette wheel selection, biased mutations (Montana and Davis, 1989), and one-point crossover. The population size was set to 80 and each individual performed 80 actions. The neural network architecture was fixed and consisted of a single layer of synaptic weights from eight input units (clamped to the sensors) to two output units (directly connected to the motors) with mobile thresholds, logistic activation functions, and recurrent connections. Synaptic connections and thresholds were coded as floating point numbers on the chromosomes. Each motor action lasted 300 ms. The fitness criterion (F) was a function of the average rotation speed of the two wheels (V), the algebraic difference between signed speed values of the wheels (DV), and the activation values of the proximity sensor with the highest activity (I):

$$F = V * (1 - \mathrm{sqrt}(DV)) * (1 - I)$$

F has three components: the first one is maximized by speed, the second by movement in a straight direction, and the third by the avoidance of obstacles. What is important to notice is that the whole evolutionary process was carried out entirely on the robot. This means that each individual network of each generation was evaluated by letting the robot move in the real environment for 80 time steps (the "brain" of each individual being sequentially injected in the same physical robot).

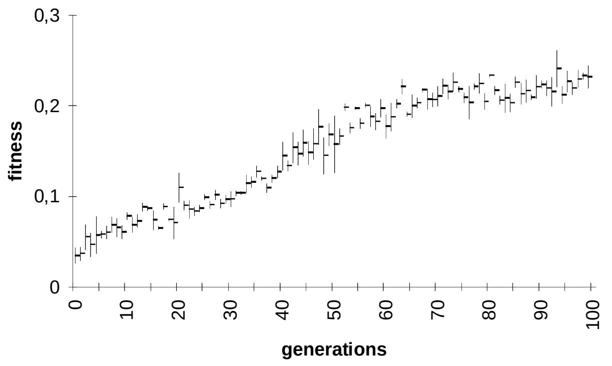

Figure 2. Best individual fitness throughout generations. Values are averaged over three runs (S.E. displayed).

Kepera genetically learned to navigate and avoid obstacles in less than 100 generations (figure 2). However, around the 50th generation the best individuals already exhibited a nearly optimum behavior. Their navigation was extremely smooth; they never bumped into walls and corners, and tried to keep a straight trajectory. This allowed them to perform complete laps of the corridor without turning back. The whole experiment lasted about 60 hours.

3. The evolution of an exploration ability by using a simulated approach.

Miglino, Nafasi, Taylor (1994) evolved recurrent neural networks to control a mobile Lego robot that should explore an open arena by using a rough simulator of the robot/environment interaction. The on-board computer used was a Miniboard 2.0 developed at the MIT Media Laboratory, Cambridge Massachusetts. It was a single board computer, optimized for controlling small DC motors and receiving data from various electronic sensors. The CPU was a Motorola 6811 micro-chip, an 8-bit microprocessor with 256 bytes of internal RAM and 12K bytes of electronically erasable programmable ROM. The robot relied on two wheels for its locomotion and had two optosensors located on the main frame. For each bout of sensory stimulus, the robot performed one of four fixed actions: a) go forward 10 cm; b) go backward 10 cm; c) turn left 45 degrees; or d) turn right 45 degrees. Because of wear and other unpredictable causes the effective action could slightly vary.

The robot was expected to explore the greatest percentage of an open arena within an allotted number of steps. The arena was 2.6 x 2.6 meters. At its center was a white-colored square, 2.0 x 2.0 meters, marked into a 20x20 grid, 10 cm per side. Surrounding this was a black border area, so that the optosensors on the robot could detect whether it was on the white or on the black surface. The lighting was provided by ordinary fluorescent room lights. The number of steps allowed was 400 and lasted about 6 minutes depending on the condition of the battery, the sequence of steps, etc.

The robot was controlled by a recurrent neural network (Elman, 1990), with 2 sensory units, 2 output units, 2 hidden units, and 1 memory unit. A simulated robot was trained in a simulated environment that was represented in a simplified way with respect to the real environment. It contained 26 X 26 cells, each representing 10 cm square, with a central white grid of 20 X 20 cells. The robot was considered as always located above the center of a single 10 cm. square. The simulated optosensors always sensed one cell ahead and behind with respect to the robot's current location. Actions in the simulated environment were represented as jumps from cell to cell.

A simple genetic algorithm (Holland, 1975; Goldberg, 1989) was used to evolve the weights for the neural network connections. The genotype of each individual in the population was represented by a vector of 17 integer numbers. Each individual was randomly positioned in the simulated environment 10 times, and, at each new starting point, was let free to move for 400 steps. Networks were scored for the number of cells touched by the simulated robot and visited for the first time. Those networks with higher scores were selected for reproduction. A population size of 100 individuals per generation was used. The top 20 individuals were allowed to reproduce by generating 5 offspring each. Mutations were introduced by randomly modifying 10% of the offspring genes. The simulation lasted 600 generations (about 3 hours using a SUN SparkStation).

The simulations showed that an efficient explorative behavior emerged throughout generations. Three different individuals representative of different phases of a particular simulation were transferred into the physical robot and tested in the real environment. Despite the fact that the trajectories of the robots in the real environment significantly differed from the trajectories observed in the simulated environment, the authors showed that the correlation between the fitness values observed in the two conditions were fairly high (0.73).

4. The evolution of a navigation ability using a hybrid (simulated/physical) approach

Nolfi, Miglino and Parisi (1994) developed neural controllers for autonomus agents that should perform a navigation task by using a hybrid approach. The robot used was Khepera (see section 2). A simulator of the interaction between such a robot and an environment similar to that described in section 2 was built. The environment was a rectangular box 60x35 cm with an obstacle of 30x5 cm placed in the center. The walls and the obstacle were made of wood and had natural wood color.

In order to build the simulator the authors sampled the enviroment by letting Khepera turn 360^{o} and by recording the sensory activations at different distances with respect to a wall (an automatic procedure that can be used to also sample other types of objects was developed). The activation level of each of the eight infra-red sensors was recorded for 180 different orientations and for 20 different distances. In addition, the authors sampled how and how much their own Khepera moved and turned for some of the 20x20 possible states of the two motors (the result of the other symmetrical states was computed without actually sampling them). These information was used by the simulator to set the activation level of the neural network inputs and to compute the displacements of the robot in the simulated environment during the first phase of the evolutionary process. The physical shape of Khepera, the environment structure, and the actual position of the robot were accurately reproduced in the simulator with floating point precision.

The neural network architecture was fixed and consisted of a feed-forward neural network with eight input units (coding the 8 infra-red sensors), 2 hidden units, and two output units (coding the state of the motors). Mobile thresholds and logistic activation functions were used. Synaptic connection and thresholds were coded as floating point numbers on the chromosomes. Each motor action lasted 100 ms. A simple genetic algorithm was used to evolve the weights for the neural networks. The genotype of each individual in the population was represented by a vector of 24 real numbers. Each individual was evaluated by randomly positioning it in the simulated environment 2 times and then leaving it free to move for 500 steps each time. The same fitness function described in section 2 was used. A population 100 in size was used and the top 20 individuals were allowed to reproduce by generating 5 offspring each. Mutations were introduced by randomly modifying 20% of the offspring genes. Noise was added to the sensory activation values.

We ran 3 experiments starting with different randomly assigned weights. The first part of the simulation, performed in the simulated environment, lasted 300 generations (about 1 hour using an IBM RISK/6000). Then the evolutionary process continued in the real environment for 30 generations.

Figure 3 shows the performances of the best network throughout generations in the simulated environment. Figure 4 shows the performances of the same networks tested in the real environment (performances of the 30 additional generations evolved in the real environment are also shown). Performance of the evolved networks significantly decreased if tested in the real environment. On the other hand, performances similar to that obtained in the simulated condition were obtained by continuing the evolutionary process in the real environment for only few generations. This means that the performance decrement in the transfer to the real robot is not due to a failure of the evolved behavioural strategies, but rather to a mismatch between the simulated and the real sensory-motor apparatus. The fast recovery rate documents that only few adjustments were needed in order to achieve a successful behaviour in the real environment.

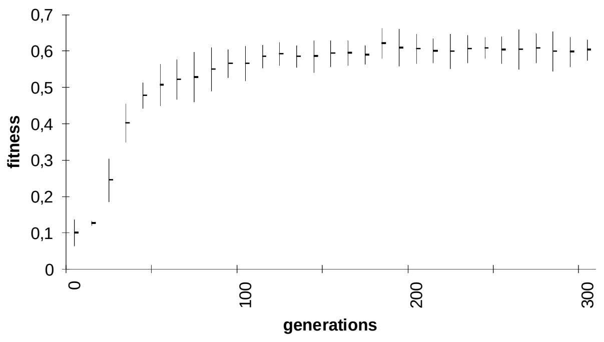

Figure 3. Performances of the best individuals throughout generations tested in the simulated environment. (S.E. displayed).

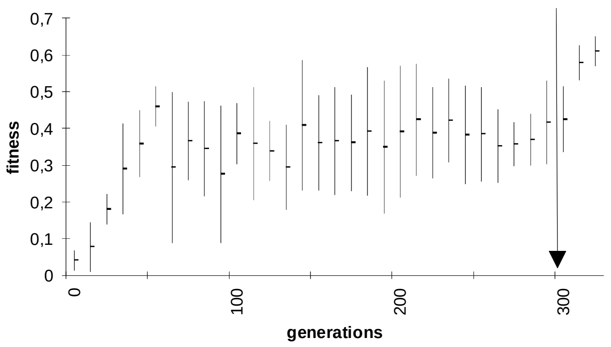

Figure 4. Performances of the best individuals throughout generations tested in the real environment. The first 300 generations have been evolved in simulated environment, the last 30 generations in the real environment (S.E. displayed).

5. Simulation versus physical approaches

The experiments described in sections 2 and 4 show that, at least in the case of simple (but not trivial) tasks, the evolutionary process can be carried out on real robots. Evolving a control system in a real environment is certainly time-consuming, but it appears to be feasible.

Nevertheless, as emerged from sections 3 and 4, computer simulations can also be useful. As several researchers pointed out (Brooks, 1992; Floreano and

Mondada, in press), there are several reasons why those who want to use simulative models to develop control systems for real robots may enconter problems:

(a) Sensor reading should not be confused, as happens in some simulative models, with the description of the world. Real sensors do not separate objects from the background, they do not operate in a stable coordinate system, and they do not give information regarding the absolute position of objects.

(b) Numerical simulations do not usually consider all the physical laws of the interaction of a real agent with its own environment, such as mass, weight, friction, inertia, etc.

(c) Physical sensors deliver uncertain values and commands to actuators have very uncertain effects, whereas often simulative models use grid-worlds and sensors which return perfect information.

(d) Different physical sensors and actuators perform differently because of slight differences in the electronics and mechanics or because of their different positions in the robot. This fact is usually ignored in simulative models.

Some of these problems may be easily eliminated by designing the simulative models carefully. For example, one should avoid using sensors or fitness functions that cannot be implemented on a real robot or that use information which is not available to the robot in the real environment (see also next section). Designing simulators based on samples of the real environment, as shown in section 4, can also avoid very difficult problems like the fact that identical sensors may respond differently. Finally, noise can be added to the simulated sensors (as shown in section 4) and introduced in the simulated actuators in order to take into account the fact that physical sensors and actuators do not perform accurately. The right amount of noise that should be introduced in order to emulate the inaccuracy of the physical sensors and actuators can even be measured by confronting the behavior of the robot in both the real and simulated environments by using different levels of noise (see Miglino, Nafasi, and Taylor, 1994). We believe that, as for the experiments described in sections 3 and 4, simulative models can be useful in developing control systems for real robots when these special solutions are taken into account.

However, one should not expect control systems which have evolved in a simulated environment to behave exactly the same as in the real environment. This is not necessary. We can be satisfied by an above-zero performance after the transfer to the real world. Once we have obtained this, the evolutionary process can be continued in the real environment for a few generations and produce perfectly adapted individuals (see section 4). From this point of view the evolution in the simulated environment can be interpreted as a selection for correlated characters and the change from the simulated environment to the real environment as a change in the environment (Prof. Charles Taylor, personal communication but see also Falconer, 1981). Different kinds of hybrid approaches may be also pursued: for example, the most promising individuals of a population evolving in a simulated environment may be tested in the real environment, or tests in the real environment can be made at given intervals during the evolutionary process.

Another important reason for using simulative models is that they can allow preliminary studies of the evolutionary process. It is well known that genetic algorithms are sensitive to the initial conditions. Different runs of the same simulation may produce solutions with different performances. Evolution in simulated environments, being usually less time-consuming than evolution in real worlds, may allow us to ascertain to what extent a specific evolutionary process is sensitive to the initial conditions and therefore what the probability would be that a limited number of simulations in real environments could produce desired performances. Similarly, simulative models can be used to set a number of important initial parameters such as selecting a good architecture for the neural network or finding the best organization of the environment to be used during the training process.

Even if it is certainly true that as the studied problems will become more and more complex it will be more and more difficult to build useful simulators, we think that at the moment, the use of simulation can still be helpful.

6. The automatic evaluation of the individuals

The development of a control system for an autonomous robot implies the evaluation of a very high number of different individuals. This fact forces to adopt an automatic way of evaluating individuals. This implies that the fitness function should compute only information that is available to the robot through its internal or external sensors. The fitness function used in the experiments described in sections 2 and 4 is a good example of that. It uses information that is available to the robot through its Infra-red sensors and through its internal sensors of the state of the motors. On the contrary, the fitness function used in the simulations described in section 3 may make the automatic evaluation of the system in the real environment difficult (unless one has some sort of device to detect the robot's exact position). In simulations all information is available and therefore the fitness function can be freely designed. On the other hand one should consider that even if the entire evolutionary process is to be performed in the simulated environment, the fact that the fitness function cannot be implemented in the physical condition can create serious limitations in evaluating the real robot performance.

Additional sensors may be dedicated exclusively to the fitness evaluation. In some experiments that we are conducting we try to evolve a robot that should stop close to objects of small sizes and ignore objects of larger sizes. In order to automatically evaluate individuals we painted the floor black around small objects and used this information for rewarding individuals that went close to small-size objects. The fitness was computed by reading the value of an infra-red sensor positioned under the robot; this value is not provided to the robot neural network. Therefore once the evolutionary process ended, we could remove the black spots on the floor without affecting the robot behavior (i.e. the black spots and the sensor under the robot are used only in order to evaluate the individuals' fitness).

7. Hardware requirements

The evolutionary approach substantially affects also the physical characteristics of the robots employed. Within the classic approach, where the control system implementation is necessarily preceded by a modelization phase, the hardware designer is faced with many important constraints. During the modelling of the interaction between the robot and the environment, the model of the robot itself plays a crucial role. Thus, the engineer must design the robot in order to make its modeling process feasible and simple enough. This leads to the choice of sensors with very linear response, actuators with a limited number of degrees of freedom (but geometrically optimal), low-noise electronic devices that can take highly precise measures, etc. Unfortunately, these systems may result as being sub-optimal and not very efficient in the real world where intrinsic noise at all levels, non-linearities, and complex shapes are the basic characteristics.

The evolutionary approach does not require the choice of a specific control system. Thus, all it is needed is some general requirement concerning the proper functioning of the sensors and of the actuators provided. Obviously, this requirement is to be taken into consideration by the designer, but it can be greatly simplified by the adoption of a greater number of devices. Such an approach would hence result in robots which are provided with more (and possibly redundant) sensors than traditional robots. However, these sensors would be basically simpler, without corrections for intrinsic non-linearities, special protections from noise, and highly precise measuring devices. It is up to the evolutionary mechanism to exploit these non-linearities or somehow amend them, and properly combine various measures to extract the information necessary for a proper functioning of the system itself. The working tools would drastically change as well. As we have already pointed out, within the classic approach the modelling phase plays an important role, whereas the analysis of the robot behavior is reduced to a confirmation of the model. On the contrary, within the evolutionary approach the modeling process can be reduced or completely avoided in order to leave more space to behavior analysis which is indeed a key point of the procedure. It is thus necessary to devise a set of new tools and methodologies that could allow a better observation, measurement, and analysis of the robot behavior. This may also facilitate the automatic evaluation and final performance monitoring.

Khepera is an attempt in this direction. Its sensory devices are very simple, but still allow a wide range of interesting experiments. In the basic version, Khepera is provided with eight proximity sensors which are based on emission and reception of infrared light. These sensors afford both the measurement of ambient infrared light, and of obstacle proximity by detection of the reflected infrared light emitted by the robot itself. These measures are not very precise, do not have linear characteristics, and strongly depend upon external factors, such as the object materials, color, illumination conditions, etc. Additional sensors can be easily added thanks to a good hardware and software modularity of the system. The miniaturization of the system itself and the environment tools developed for this experimental platform are clearly designed in the direction of the analysis, rather than toward the classic design approach. Khepera has been conceived and designed in order to be easily used on the top of a desk, close to a workstation and connected to it via a serial line that also supplies the electricity to the robot. This configuration affords optimal experiment conditions by allowing easy monitoring of the robot, the real environment, and the computer. Such a configuration is particularly well-suited for experiments in evolutionary robotics where the robot may display "pathological" form of behaviors for long periods of time (initial generations), e.g. crashing into walls or simply pushing against obstacles. With some simple precautions (e.g., walls in polistyrene) and thanks to physical laws according to which homotetically reduced objects offer greater mechanical robustness, these experiments can be carried out without problems.

8. Conclusions

A methodology has not yet been well established in order to evolve control systems for autonomous robots. In this paper we showed three different examples of the application of evolutionary techniques to real robots. The results of our experiments showed that evolving control system in real environment is feasible even if it is certainly time-consuming (see section 2). We showed that also simulations can be useful in evolving control systems for real robots. In some cases even a rough model of the real robot and environment can be enough to evolve control systems that can then be transferred to the physical robot (see section 3). There are ways of designing simulative models that significantly reduce the discrepancies between the simulated and real conditions. In particular we showed how to design the simulative model by sampling the real environment through the real sensors and actuators of the physical robot may result in software models that approximate much better the real environment condition (see section 4). We believe that one should not expect the control system which evolved in the simulated environment to behave exactly the same in the real environment. We rather believe that a hybrid approach in which part of the evolution process is performed in the simulated environment and part in the real one would be more fruitful (see section 4).

Another important issues is whether or not these approaches can be applied to more complicated tasks and which tasks are particolarly suited to them. We are exploring different directions. In a current experiment Floreano and Mondada (1994) provide the environment with a zone where the robot battery gets automatically charged (a simulated charge based on a hardware prototype under test) and an oriented light source. The robot is also provided with a few more sensors (light sensors), but the fitness function is exactly the same as the one employed for obstacle avoidance. The difference with respect to the experiment described in section 2 is that the robot can recharge its battery and thus prolong its own life if it passes over the charging zone. By employing the same evolutionary procedure, the robot learns to keep itself "alive" by periodically returning to the charging zone. This emergent homing behavior is based on the autonomous development of an internal topographical map that allows the robot to choose the appropriate trajectory as function of its location and of its remaining energy. In another current experiment the authors try to address the issue of object recognition for grasping using the miniature robot Khepera with an added gripper module. The environment is a surface with a number of objects and obstacles. The objects, as in natural situations, feature different shapes and sizes. The robot is expected to autonomously learn to approach only those objects that can be physically grasped by its own gripper module.

Acknowledgments

We would like to thank Domenico Parisi and Charles Taylor who provided important comments. Dario Floreano has been partially supported by C.N.R grant n. 93.01065.PF67 (co-ordinator: Walter Gerbino), Francesco Mondada by the Swiss National Research Foundation (project n. PNR23), and Stefano Nolfi and Orazio Miglino by P.F. "ROBOTICA", C.N.R., Italy.

References

Ackley, D. H., M. L. Littman, 1991. Interactions between learning and evolution. In *Artificial Life II*, edited by C. G. Langton, J. D. Farmer, S. Rasmussen, C. E. Taylor. Addison-Wesley. Reading, Mass.

Brooks, R. A. 1991. New approaches to robotics. *Science* **253**:1227-1232.

Brooks, R. A. 1992. Artificial life and real robots. *In Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* edited by F. J. Varela, P. Bourgine. Cambridge, MA: MIT Press/Bradford Books.

Cliff D. T., I. Harvey, P. Husbands. 1993. Explorations in Evolutionary Robotics. *Adaptive Behavior* **2**: 73-110.

Collins R., D. Jefferson. 1991. Representations for artificial organisms. In *Proceedings of the Simulation of Adaptive Behavior*, edited by J. A. Meyer, S. Wilson. Cambridge, MA: MIT Press/Bradford Books.

Dorigo M., U. Schnepf. 1993. Genetis-based machine learning and behavior based robotics: a new syntesys. *IEEE Transaction on Systems, Man and Cybernetics*, **23**:141,153.

Elman J. L. 1990. Finding structure in time. *Cognitive Science, 2: 179-211.*

Falconer D. S. 1981. *Introduction to Quantitative Genetics*, Longman, London.

Floreano D., F. Mondada. 1994. Emergent homing behaviour in a mobile robot. *Technical Report* LAMI n. DF94.14I, Swiss Federal Institute of Technology, Lausanne.

Floreano D., F. Mondada. In press. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In *From Animals to Animats 3: Proceedings of Third Conference on Simulation of Adaptive Behavior*, edited by D. Cliff, P. Husbands, J. Meyer, S. W. Wilson. MIT Press, Bradford Books.

Goldberg D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Mass.: Addison Wesley.

Holland J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Koza J. R. 1990. Genetic Programming. A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. *Technical Report* STAN-CS-90-1314. Stanford University Computer Science Dept.

Miglino O., R. Pedone, D. Parisi. 1993. A noise Gene for Econets. In *Proceedings of Genetic Algorithms and Neural Networks*, edited by M. Dorigo, Reading, Mass.: Addison Wesley.

Miglino O., K. Nafasi, C. Taylor. 1994. Selection for Wandering Behavior in a Small Robot. *Technical Report*. UCLA-CRSP-94-01. Department of Cognitive Science, University of California, Los Angeles.

Mondada F., E. Franzi, P. Ienne. 1993. Mobile Robot miniaturisation: A tool for investigation in control algorithms. In: *Proceedings of the third International Symposium on Experimental Robotics*, Kyoto, Japan.

Montana D., L. Davis. 1989. Training feed forward neural networks using genetic algorithms. In *Proceedings of the*

Eleventh International Joint Conference an Artificial Intelligence edited by N.S. Sridharan. San Mateo: Morgan Kaufmann.

Nolfi, S., O. Miglino, D. Parisi. 1994. Phenotypic plasticity in evolving neural networks: Evolving the control system for an autonomous agent. *Technical Report* n. PCIA-94-04, Institute of Psychology, C.N.R., Rome.

Parisi, D., F. Cecconi, S. Nolfi. 1990. Econets: Neural networks that learn in an environment. *Network* **1**:149-168.

Parisi, D., S. Nolfi. In press. How learning can influence evolution within a non-Lamarckian framework. In *Plastic Individuals in Evolving Populations*, edited by R. K. Belew, M. Mitchell. SFI Series, Addison-Wesley