Sample Contributed Book

Sample Contributed Book

Edited by J. SMITH

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

Contents

1

Evolutionary Mobile Robotics

DARIO FLOREANO

Microcomputing Laboratory, Swiss Federal Institute of Technology at Lausanne

1.1 Introduction

Robotics represents an interesting application domain for evolutionary computation because robot parameter optimization often implies search of high-dimensional spaces characterized by nonlinear and non-differentiable relationships between variables, which make analytical solutions quite difficult.

Evolutionary algorithms are time-expensive procedures where most of the time is spent on evaluating the fitness of each individual in a population for several tens or hundreds of generations. In general, optimization speed is solely determined by the performance of the computer on which the algorithm runs. However, in robotics each individual string in the population must be tested on the robot itself, which results in a drastic time bottleneck. Resorting to software simulation of the robot is not always a suitable choice because the evolved solution will be biased by the description level and inaccuracies of the simulator. Furthermore, almost all robots are subject to wear, friction, malfunctioning, and other physical factors which can never be sufficiently taken into account in a simulation [Bro92], but which might indeed be relevant for the control parameters under optimization. On the other hand, evaluating several individuals on the same robot requires appropriate technological solutions and reliable hardware.

There are several situations where robotic parameter optimization can be done without resorting to time-consuming genetic algorithms. For example, if the robot is expected to operate in highly controlled environments and one knows in advance exactly what actions should be performed for each sensory information, the control parameters might be approximated by performing gradient descent on the error surface

Genetic Algorithms in Engineering and Computer Science
Editor D. Quagliarelli

given by the discrepancies between the correct actions and the robot actions. This procedure can be used for most robots employed in today's manifacturing plants.

1.1.1 Autonomous Mobile Robotics

The scenario is quite different in the case of *mobile* and *autonomous* robots. These are machines that are required to carry out a task in partially unknown and unpredictable environments. Such tasks typically require exploration, orientation, and goal-directed navigation abilities. Here, it is not always possible to define in advance what a correct action is for every possible sensory pattern, for at least three reasons: a) sensors might return continuous, noisy, and ambiguous values, which make it difficult to create table of sensor-motor associations; b) the same sensory configuration might offer several possible viable actions to choose from; c) the choice of particular actions might depend on the sequence of previous sensory patterns and on other parameters of the robot and of the environment (for example, battery charge level, location in the environment, amount of work left to be carried out, etc.).

The classical approach of task analysis, decomposition, and planning tends to generate huge, slow, and brittle control systems when applied to mobile robotics [Bro90]. Modern approaches, instead, advocate use of a different methodology, often named Behavior-Based Artificial Intelligence [Bro91, Mae93], a bottom-up approach that puts emphasis on the development of control systems while the robot interacts with its own environment. Learning plays an important role within the Behavior-Based approach. Rather than changing the control program by hand (in a more or less systematic way), the robot is equipped with some learning mechanism that autonomously modifies the control parameters with regard to the mission to be achieved and on the basis of previous experiences. For example, a robot that is expected to navigate around an environment without hitting obstacles, could strengthen or weaken associations between sensory and motor patterns as it moves around depending on whether an obstacle has been hit or a certain distance has been safely covered.

A common technique is Reinforcement Learning, whereby input-output (sensor-motor states) associations are changed on the basis of an external reinforcement signal (usually a negative value stands for a bad action and a positive value for a good action) which can be provided by the environment itself (e.g., a wall has been hit) or defined by the human user for a certain task. The goal of the robot is to learn to produce those actions that maximize the sum of all reinforcements from the initial location to the goal position. Reinforcement Learning (RL) algorithms have been applied to neural networks [BSA83] and used for mobile robots (e.g., [MC92, Mil96]) in order to learn simple navigation tasks. Since most of these algorithms require storage (or approximations) of expected future reinforcement for every sensory and motor pairs, it is often necessary to discretize states in order to reduce computational costs and allow the robot to visit *all* states (some improvements on this problem have been proposed; e.g., [Mil97]). Furthermore, it is important to carefully choose an appropriate reinforcement program, that is when, which, and how often a reinforcement value is provided to the robot.

More recently, evolutionary algorithms have also been applied to mobile robots for learning navigation abilities of various types. An evolutionary algorithm is to some extent similar to an RL algorithm because the definition of the fitness function is

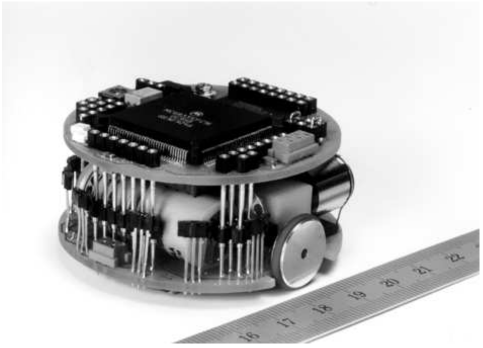

Figure 1.1 Khepera^{TM}, a miniature mobile robot. The ruler is in centimeters.

similar to the definition of a reinforcement program, but its functioning is quite different. Whereas RL performs gradient ascent on the cumulative reinforcement value, evolutionary algorithms are based on blind search and selection. *Evolutionary Robotics*, the name commonly used to indicate this approach [CHH93], is a powerful method that is well worth long training periods because artificial chromosomes can potentially encode and co-evolve several aspects of the robot, such as control architecture, developmental rules, morphological aspects, and learning rules. Furthermore, evolutionary algorithms have also been used to evolve reinforcement programs for learning neural controllers [AL92].

In this chapter, I will provide a survey of evolutionary methods and experiments on real mobile robots. In section 1.2 I will start with a simple experiment that will serve as an introduction to the methodology, and then describe experiments that will illustrate scalability to more complex environments (section 1.3), behaviors (section 1.4), and sensory information (section 1.5). I will put emphasis on the fact that evolutionary algorithms do not require much constraints and pre-designed solutions to start with; in fact, complex behaviors do evolve with simple fitness functions to cope with complex environments and missions. This will become particularly evident in the co-evolutionary experiments described in section 1.6 where competing robots are evolved in parallel. In section 1.7 I will give a short overview of evolvable hardware in mobile robotics, that is the effort of evolving not only the control program of the robot, but also the circuit design and its morphological configuration. Finally (section 1.8), I will discuss some of the advantages, limitations, and future directions of research.

1.2 Evolutionary Robotics: A Simple Illustrative Example

In order to perform evolutionary training on mobile robots, it is important to have reliable hardware and a long-lasting power source. Most of the experiments described in this chapter have been carried out on Khepera ^{TM}, a miniature mobile robot initially developed at our laboratory [MFI93] at the Swiss Federal Institute of Technology in Lausanne and currently distributed by the spin-off company *K-Team SA*. Khepera (figure 1.1) is a circular robot: it has a diameter of 55 mm, it is 30 mm high, and its weight is 70 g. The robot is supported by two wheels and two small Teflon balls placed under its platform. The wheels are controlled by two DC motors with an incremental

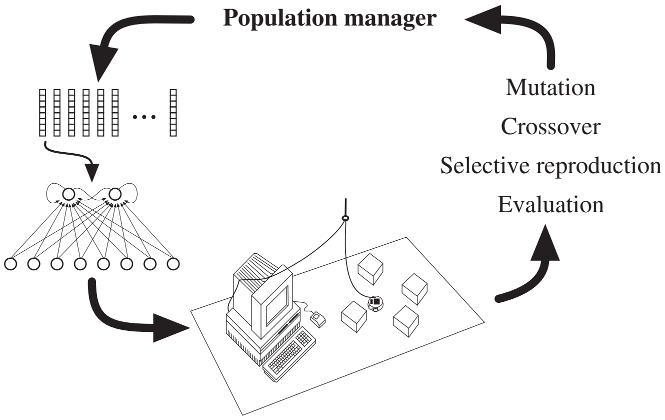

Figure 1.2 Method employed for evolutionary training on a single robot.

encoder (12 pulses per mm of robot advancement) and can rotate in both directions. In its basic version it is provided with eight infrared proximity sensors placed around its body (six on one side and two on the opposite one) which are based on emission and reception of infrared light. Each receptor can measure both the ambient infrared light (which in normal conditions is a rough measure of the local ambient light intensity) and the reflected infrared light emitted by the robot itself (for objects closer than 4-5 cm). These measures do not have linear characteristics, are not filtered by correction mechanisms, and depend on a number of external factors, such as the surface properties of objects and the illumination conditions. Several new single sensors and additional modules (such as a vision module and a gripper module) can be easily plugged-in on the top of the robot at any time.

In our experiments the robot was attached to a Sun SPARCstation via a serial line by means of an aerial cable and specially designed rotating contacts. All low-level processes –such as sensor reading, motor control, and other monitoring processes– were performed by the on-board micro-controller, while other processes (controller activation and genetic operators) were managed by the Sun CPU. The cable was used to supply power and communicate with the robot in order to keep full track of the robot behavior and development by exploiting the data-storage capabilities of the workstation and the special software for on-line behavior monitoring and analysis. An external positioning laser device was employed for *post-training* analysis of the evolved control systems: the robot was provided with an additional module for capturing the light signal emitted by the laser device and computing its own absolute position. This computation was carried out by a separate processor placed on the additional module and the result was then passed to the workstation (but *not* to the robot controller) where special software was used for automatic on-line analysis and display of the trajectories along with sensors, motors, and controller states. The positioning module, which could be easily added and removed, did not affect the robot motion and sensor activation. Such tools allowed us to perform neuroethological observations of the robot during normal operating conditions by relating the behavior displayed with the internal functioning of the controller.

As in almost all the experiments in evolutionary robotics, the controller was

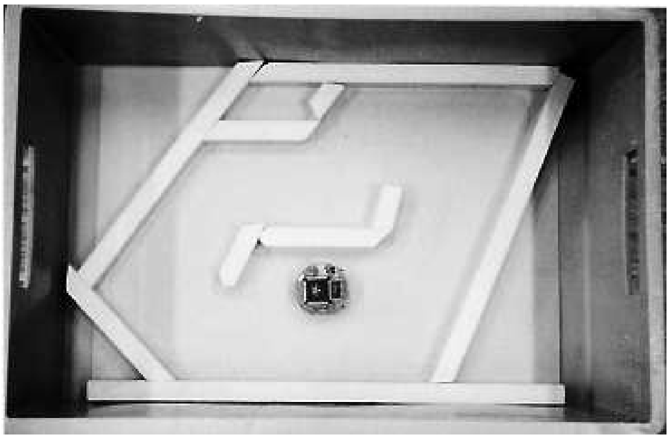

Figure 1.3 The environment employed for evolving navigation and obstacle avoidance (from [FM94]).

implemented as an artificial neural network with sigmoid units and recurrent connections. Neural networks have interesting properties for this application domain because they are noise tolerant, have intrinsic generalization capabilities, potentially display dynamical characteristics, are universal approximators, and can be equipped with a variety of learning algorithms in addition and in conjunction with evolutionary algorithms. The evolutionary procedure employed in the experiments consisted in applying a simple genetic algorithm with linear fitness scaling [Gol89] to the synaptic weight values (including the neuron thresholds) of the neural network that controlled the robot (figure 1.2). Given the small size of the networks used and the fixed architecture, the synaptic weight values were individually coded as floating-point numbers on the chromosome. Each chromosome in the population had the same constant length corresponding to the number of synaptic connections and neuron thresholds in the network. An initial population of individuals was created by assigning to each gene in the chromosomes a new value drawn from a uniform random distribution of continuous numbers within a small positive and negative range. Each individual, in turn, was decoded into the corresponding neural network, the input units were attached to the eight infrared sensors of the robot and two output unit activations were directly used to set the velocity of each of the two wheels. The robot was left free to move as a result of the activity generated by the neural network while its performance was recorded and accumulated according to a pre-designed fitness function. Each robot could move for a limited number of actions, each lasting a few hundred milliseconds.

This simple experiment was aimed at explicitly evolving the ability to perform straight navigation while avoiding the obstacles encountered in the environment. The robot was put in an environment consisting of a sort of circular corridor whose external size was approx. 80x50 cm large (figure 1.3). Since the corridors were rather narrow (8-12 cm), some sensors were slightly active most of the time.

The fitness criterion Φ was described as a function of three variables, directly

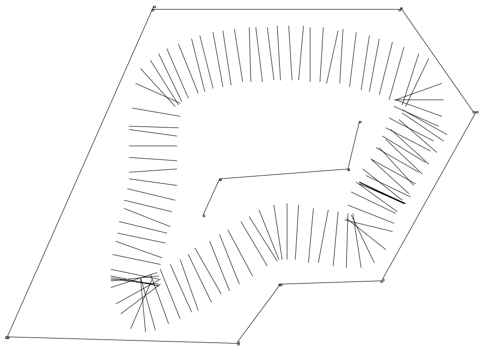

Figure 1.4 The trajectory performed by one of the evolved robots. Segments represent successive displacements of the axis connecting the two wheels. The direction of motion is counter-clockwise (from [FM96a]).

measured on the robot at each time step, as follows,

$$\Phi = V\left(1-\sqrt{\Delta v}\right)(1-i) \tag{1.1}$$

$$\begin{array}{ccccc} 0 & \leq & V & \leq & 1 \\ 0 & \leq & \Delta v & \leq & 1 \\ 0 & \leq & i & \leq & 1 \end{array}$$

where V is a measure of the average rotation speed of the two wheels, Δv is the algebraic difference between the signed speed values of the wheels (positive is one direction, negative the other) transformed into positive values, and i is the activation value of the proximity sensor with the highest activity. The function Φ has three components: the first one is maximized by wheel speed (without regard to direction of rotation), the second by straight direction, and the third by obstacle avoidance.

Khepera learned to navigate and avoid obstacles in less than 100 generations, although around the 50th generation the best individuals already exhibited an almost optimal behavior. Their navigation was very smooth, they never bumped into walls and corners while trying to keep a straight trajectory. They performed complete laps of the corridor without turning back (figure 1.4).

Each fitness component was necessary to develop this behavior. Without the first component, a robot standing still far from a wall would achieve maximum fitness. Without the second component, maximum fitness could be easily achieved by fast spinning in the same place (wheels turning at maximum speed in opposite directions) far from walls. Finally, without the third component, evolution would develop robots moving straight (frontally or backward) at maximum speed until they crashed against a wall. Although the fitness function was accurately designed to evolve the desired behavior, a number of interesting low-level aspects of the controller evolved as a side effect of the interaction between the robot and the environment. For example, despite the fact that the robot could theoretically learn to move either forward of backward (because it has a circular shape), the best individuals in all our runs developed a frontal direction of motion, corresponding to the side where there are more sensors. Robots moving backward sometimes were stuck into obstacle which could not be perceived and, since the first fitness component would give values close to zero, these individuals would not be selected for reproduction. Similarly, the best individuals developed an optimal cruising speed which was not the maximum available, but was perfectly adapted to the refreshing rate of the sensors. Robots moving faster would crash into obstacles before having detected them and would therefore soon disappear from the population. Finally, the best individuals developed appropriate recurrent connections at the output layer which worked as a tie-breaking mechanism when symmetric stimulation of sensors on both sides would generate equal and opposite signals to the wheels. Such robots would never get stuck in corners while retaining normal navigation abilities in all other situations.

1.3 Environmental Influence

In the experiment described above, the fitness function was carefully selected to evolve the desired navigation behavior. Although designing a fitness function is much easier than designing a functional controller, one might wonder whether the complexity of the fitness function is monotonically related to the complexity of the desired behavior. If this is the case, the evolutionary approach would quickly become impracticable as one wishes to evolve more complex behaviors or employ a higher number of sensors and actuators. A related issue concerns the searching abilities of genetic algorithms for complex fitness functions. If we assume that a complex fitness function (that is, a function of several variables characterized by nonlinear relationships) corresponds to a complex fitness surface (that is, a surface displaying several peaks surrounded by sharp minima and lower peaks),[1] then genetic search of the maxima on this landscape might not be very efficient.

In this section, I intend to show that the evolution of more complex behaviors does not necessarily require more complex fitness functions, but is rather enabled by the possibility of a richer interaction between the robot and the environment. In this second experiment, the Khepera robot was equipped with a simulated battery that lasted only 20 seconds (for the purpose of accelerating evolutionary training). The robot was put in a rectangular environment with a (simulated) battery charger in one corner under a light source oriented towards the environment (figure 1.5). The floor was white except for the area with the battery charger which was painted black. The robot was equipped with an additional sensor underneath the platform measuring the floor brightness and two light sensors, one on the front and one on the back. The neurocontroller was a multilayer perceptron with a set of recurrent connections among the hidden units. The input units were clamped to the eight infrared sensors, to the two light sensors, to the floor sensor, and to a sensor of battery charge. Two output units controlled the wheel of the robot. The fitness function was a simplified version

[1] As a matter of fact, this is not necessarily true.

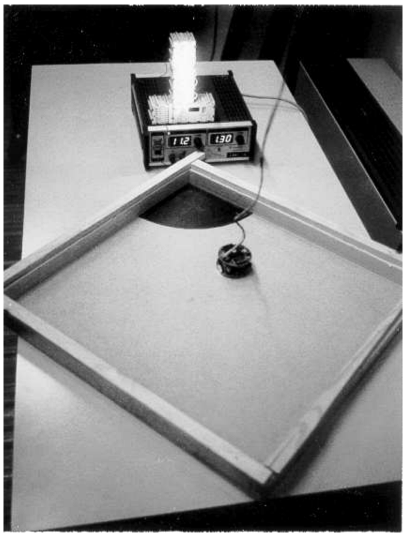

Figure 1.5 The environment of the experiment on battery recharge. The light tower is positioned in the far corner over the recharging area which is painted black. There are no other light sources in the room (from [FM96a]).

of 1.1,

$$\Phi = V(1 - i) \tag{1.2}$$

$$\begin{array}{ccccc} 0 & \leq & V & \leq & 1 \\ 0 & \leq & i & \leq & 1 \end{array}$$

without the middle component responsible for straight trajectories in the previous experiment. As in the previous case, the fitness values were computed after each robot action (300 ms) and accumulated during the "life" of the individual. Each individual started with a full battery, corresponding to about 50 actions. However, if the robot happened to pass on the recharging area, the battery was immediately recharged prolonging the life span for 50 more actions (all individuals were "killed" after a maximum of 150 actions to leave space to the next individual in the population). While on the recharging area, the fitness function returned values close to zero (because the robot was very close to the walls and the infrared sensors thus returned very high values, setting to zero the second component of the fitness function).

Figure 1.6 shows the behavior and neural activity of the best individual at generation 240. As most of the best individuals in the final generations, it navigated around the environment, avoiding the walls and, only when the battery was almost discharged, it returned to the recharging area. Once on the recharging area, the robot quickly turned on itself and resumed navigation in the arena. This behavior was maintained indefinitely or until we stopped its life. In a further set of tests [FM96a], it was shown that the internal hidden units were specialized for different aspects of the evolved behavior. While one unit was in charge of performing reactive obstacle avoidance,

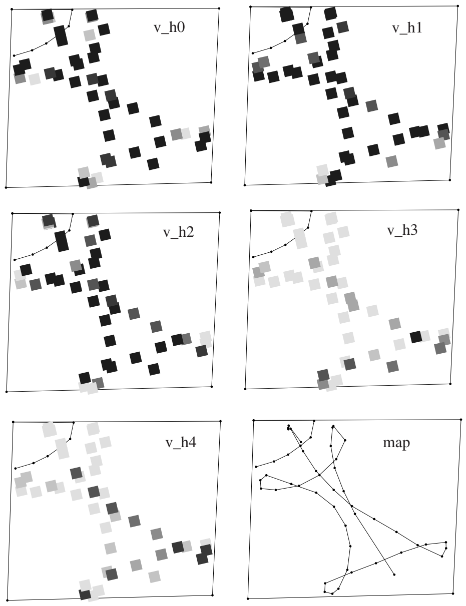

Figure 1.6 Visualization of the hidden node activations (five hidden nodes) while the best robot of the final generation moves in the environment. Darker squares mean higher node activation. The robot starts in the lower portion of the arena. The bottom-right window plots only the trajectory. The recharging area is visible in the top left corner (from [FM96a]).

another one developed a spatial representation of the environment which, combined with information on battery charge, allowed the robot to compute exactly when to initiate the homing behavior depending on its distance from the recharging area and the residual energy level.

The basic idea behind this experiment was that of making the environment and the robot characteristics more complex, thus increasing the set of possible interactions, while decreasing the constraints on the fitness function. It should be noticed that this approach is quite different (and much simpler) from what one would typically do within a reinforcement learning approach. In the latter case, the desired behavior would be decomposed in sub-behaviors, such as obstacle-avoidance, light-following, planning (to decide when to switch behavior depending on battery charger), and an appropriate reinforcement signal would be allocated for the completion of each behavior. Here, instead, the global behavior emerges as a consequence of the attempt to maximize life duration; the designer must not explicitly reinforce the system when it reaches the battery charger, or analyze and decompose the desired behavior. The ability to locate and return over the recharging area when necessary is not treated as one of the main goals to be achieved; rather, it is only one possible behavioral strategy that could be developed in order to maximize life duration and therefore fitness values. The results reported here and in the previous section show that the environment can play an important role within the evolutionary approach in shaping appropriate abilities and behaviors.

1.4 Behavior Decomposition and Controller Modularity

In the experiments described above, the evolved controller displayed some degree of functional specialization. Some hidden units were active only when the robot avoided obstacles, whereas other units were active during re-orientation and homing for battery recharge (figure 1.6) [FM96a]p. 402. At a *post-hoc* analysis, one might be tempted to describe this organization as an emergent behavioral decomposition of the global behavior in two simple modules corresponding to obstacle avoidance and homing. Whether or not such a description fits the actual behavior, what is important here is that the decomposition was not pre-designed and externally imposed, but rather evolved out of a monolithic architecture as a computationally-efficient strategy to carry out the global behavior.

Despite this result, the choice of immutable monolithic architectures is unlikely to scale well when the required set of behaviors increases. To illustrate this point, let us consider the case of a neural controller with a set of fully-connected nodes. Some nodes will be clamped to the sensors, others to the actuators, and the remaining will do some internal processing. Let us imagine that there are two output units, one that decides the direction of motion of the robot and the other that activates a gripper. In this architecture the change to a single synaptic weight, caused for example by the mutation operator, is likely to affect both the postsynaptic node and all the other nodes that receive activation from it. Therefore, a mutation which is beneficial for motion might badly affect the gripping abilities. In other words, such a controller would be characterized by high epistatic boundaries, that is the fitness contribution of each gene would depend on several other genes in the chromosome [KJ92].

Several authors [Mae92, MC92, DS93] have decided to start with a controller composed of several modules, each corresponding to a simple behavior (or "reflex"), which are activated by an action selection mechanism. Both the action selection mechanism and the individual modules can learn while the robot interacts with the environment. Stefano Nolfi, working at the National Research Council in Rome, has shown that evolved modularity can outperform pre-designed modularity by allocating resources to modules in ways which do not necessarily reflect our logic and linguistic description of behaviors, but rather take into account the sensory motor characteristics of the robot and of the controller type.

Nolfi [Nol97] has employed a Khepera robot with a gripper module (figure 1.7) for keeping an arena clear of garbage. The robot mission was that of finding, collecting, and throwing garbage outside the walls of the arena. This task was chosen because it is quite complex and amenable to be decomposed in several sub-tasks (as other authors have often done; e.g., [CDB96]). For example, a first decomposition could be: a) look for garbage and avoid walls; b) look for walls and avoid remaining garbage. One might then decompose each of these modules into other submodules necessary for correctly grasping and releasing objects, and so forth.

Nolfi has evolved and compared three types of architectures: a monolithic network, a pre-defined modular network, and a network with *emergent* modularity (figure 1.8). The network with a pre-defined architecture was composed of two modules reflecting the decomposition described above: one module was active when the gripper was

Figure 1.7 The Khepera robot with the gripper module.

empty (behavior a) while the other module was active when the gripper was full (behavior b). For the architecture with emergent modularity, the decision of which module was active at any one time was taken by the winner between neurons competing for activating their own module. The synaptic weight values of the competing neurons were co-evolved with the synaptic weight values of the motor neurons in each module. In order to speed up evolutionary training, he evolved all the neural networks in simulation and assessed their performance by testing the evolved best individuals on the real robot. The genetic method employed was similar to that described in section 1.2 and the fitness function was based on the number of objects correctly released outside the arena and on the number of objects correctly lifted (this second component had a low weighting). His results showed two main points. The first point was that the networks with emergent modularity reported on average the highest fitness values during training and largely outperformed all the other architectures when tested on the real robot. The second point was that the evolved module specialization and task allocation did not reflect different environmental situations, as a human designer would do, but rather how the world looks like from the point of view of the robot. By analyzing how each module gained control with respect to the patterns of sensory information, it was shown that different modules were allocated when different motor actions were required for similar sensory patterns. In other words, the evolved modularity was not based on a *distal* description (from the point of view of the observer) of the environment, but on a *proximal* description closely related to the characteristics of the sensory motor mapping from the point of view of the robot.

Since a proximal decomposition of behaviors cannot be mapped into a distal decomposition, identification, Nolfi argued that hand-crafting, and combination of several modules for achieving complex behaviors will become unfeasible or produce sub-optimal results, hinting at the fact that co-evolution of modules and action

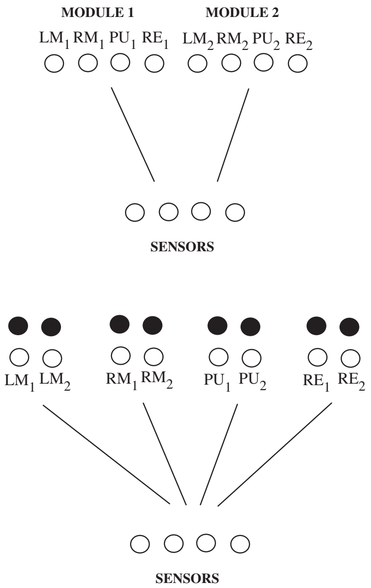

Figure 1.8 Two types of architectures evolved in Nolfi's experiments. Top: pre-designed modular architecture. Bottom: emergent modular architecture. LM = left motor; RM = right motor; PU = pick-up; RE = release (adapted from [Nol97]).

selection mechanisms might be a viable solution.

1.5 Visually-Guided Behaviors

The sensory system of the robots that we have considered so far is very simple and comparable in complexity to that of bacteria capable of only a few behaviors, such as chemotaxis, phototaxis, and obstacle avoidance. In nature, more interesting behaviors are supported by more sophisticated sensory systems, such as vision. The visual system, be it that of a fly or that of a cat, is a multi-layer organ based on a compact set of light receptors geometrically organized on the epidermic surface of the animal and locally relayed to a few layers of specialized neurons which extract information to be passed further on to other processing, multi-modal, and/or motor areas of the nervous system. The fact that visual systems have been discovered and re-discovered by natural evolution along different phyologenetic pathways indicates that vision is an efficient system to collect useful information about the environment in a planet bathed in light as ours is.

The realization of artificial visual systems has been one of the most challenging research enterprises since the very beginning of cybernetics. The main difficulty associated with vision is not so much its hardware realization, but how to use the millions of analog values that are generated several times a second by a retina or a CCD camera. A classic approach regards the visual system as an input device for pattern recognition. Much of the research in image processing is indeed concerned with segmentation of objects from background, discounting environmental illumination, binding together different attributes of the same object, etc. The more or less explicit goal of such enterprise is that of devising a set of computational operators capable of building a low-dimensional and faithful representation of the external world [Mar82]. Typically, these operators tend to be brittle (in other words, they are based on fine tuning of a set of parameters for a well-defined and specific objective), they have high computational costs and memory requirements, and they are slow. For this reason, most algorithms run off-line, that is they work on a collection of pre-collected static images. It is clear that this is a serious problems if one wishes to equip a mobile robot with a visual system and integrate it with the control system. But, do robots (and, for that matter, animals too) really wish to extract a representation of the environment upon which to decide what to do?

At the end of the seventies, J. J. Gibson [Gib79] warned against the dangers of an object-centered interpretation of vision and argued that visual processing is above all a viewer-centered operation. He showed that much information is implicitly given in the optical flow if one considers the visual scene from the point of view of the observer. Although Gibson's ideas were positively welcome, not much changed in the mainstream approach to visual processing (except perhaps for research on motion perception). I believe that there are two main reasons why things did not change. The first reason is that most people still think that segmenting an object from the background and creating a representation of it is a fundamental issue, although they are conscious that more attention must be paid to the ecological settings in which images are taken and to the natural sequence of images. The limiting factor here is computational and time resources because traditional algorithms cannot cope with the richness and continuous flow of natural images. The second reason is the lack of proper tools for understanding and reproducing biological vision. Almost all image collections used for evaluating today's algorithms are snapshots taken with a standard camera. When temporal sequences are available, these are translations, rotations, or zooms, which are quite different from the sequence of images sampled during saccadic exploration of a visual scene. Above all, artificially-taken images fundamentally differ from those available to biological organisms because *they are not a consequence of the motor actions* which in turn were taken as a consequence to them. In other words, in order to understand biological vision and build efficient machines one has to consider the sensory-motor loops within the larger behavioral units of an agent that interacts with its own environment.

Autonomous mobile robots represent an ideal tool to study and test theories and models of visual processing within an ecological approach. Evolutionary computation, in particular, is a suitable technique to discover both what are the important bits of visual information and how to use them to carry out a certain mission in the environment because both aspects can be co-evolved on the same chromosomes and at the same pace. Harvey *et al.* at the COGS group of the University of Sussex have carried out the first explorations on the evolution of visually-guided behaviors [HHC94].

Instead of using wheels, their robot was suspended within a rectangular environment and could be rotated and moved along the x, y coordinates. The robot had a cylindrical shape and was equipped with an obstacle detection mechanism and a wide-angle CCD camera with a 60 deg visual field. The output of the camera and of the obstacle detection device were transmitted to an external computer through a frame-grabber.

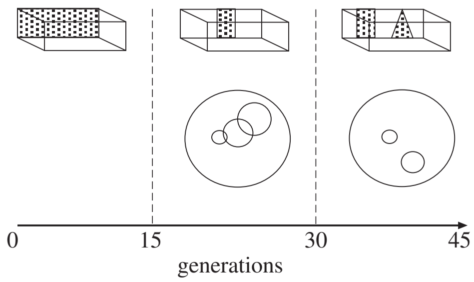

Figure 1.9 Incremental evolution of visually-guided behaviors. Top: Targets (in the experiments the walls were black and the targets white) against one wall of the arena. Bottom: Evolved visual morphologies of the best individuals at the end of each condition (no details available for the first condition) (adapted from [HHC94]).

These complex settings were devised to allow long evolutionary experiments. However, it is clear that one could do exactly the same thing using a Khepera robot with a CCD camera module and the rotating contacts described in section 1.2 (as a matter of fact, the Khepera robot has similar shape, size, motion dynamics, and a CCD module with rotating contacts –including video output– is now available too). The goal of the authors was that of co-evolving a neural controller along with the *morphology* of the visual system in order to develop visually-guided behaviors.

The genotype of each individual was composed of two chromosomes, one encoding the morphology of a neural network with a fixed input and output layer, and the other encoding the position and size of three receptive fields placed on the image (up to 256 such receptive fields could be positioned on the original image). Each input neuron received the average grey level from 25 pixels uniformely scattered across the corresponding receptive field. Two pairs of output neurons controlled the "virtual wheels" of the robot. Harvey *et al.* used an incremental approach which consists in evolving more complex behaviors from already evolved simpler behaviors [Har93]. An initial population was evolved from scratch to approach a white wall, 150 cm wide and 22 cm high (the remaining three walls of the arena were dark) for approximately 15 generations. The fitness function was the inverse of the distance from the wall. Further 15 generations, starting from the last evolved population, were carried out on a narrower white target (22 cm wide) positioned against a wall. Finally, evolutionary training was continued for another 15 generations on geometrical shapes. During this latter stage, a white isosceles triangle (21 cm wide at the base and 29.5 cm high) and a white rectangle (21 cm wide and 29.5 cm high) were positioned against one wall; the fitness function was designed in order to approach the triangle and avoid the rectangle.

Despite the simplicity of the visual information available, a set of controllers were evolved at the end of each evolutionary stage which could generate visually-guided approach of the selected target. A detailed analysis of the co-evolved visual morphology and network architectures clearly indicated that successful behaviors were based on a combination of smart positioning of receptive fields on the visual field (figure 1.9) and of a set of behaviors aimed at actively "searching" for relevant features of the visual scene. Furthermore, the best individual evolved at the end of the second stage were capable of following small white targets moved around the arena by the experimenters.

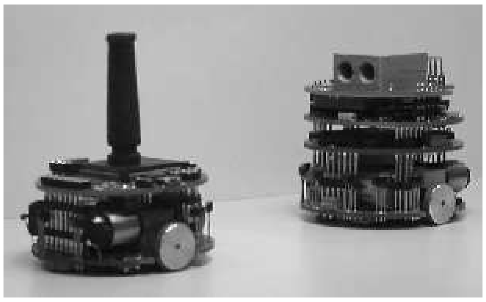

Figure 1.10 Right: The Predator is equipped with the vision module (1D-array of photoreceptors, visual angle of 36°). Left: The Prey has a black protuberance which can be detected by the predator everywhere in the environment, but its maximum speed is twice that of the predator. Both Predator and Prey are equipped with 8 infrared proximity sensors (max detection range was 3 cm in our environment).

1.6 Co-Evolution of Competing Robots

Competitive co-evolution has recently attracted considerable interest in the community of Artificial Life and Evolutionary Computation. In the simplest scenario of two co-evolving populations, fitness progress is achieved at disadvantage of the other population's fitness. Although it is easy to point out several examples of such situation in nature (e.g., competition for limited food resources, host-parasite, predator-prey), it is more difficult to analyze and understand the importance and long-term effects of such "arms races" on the development of specific genetic traits and behaviors. An interesting complication is given by the "Red Queen effect"[2] whereby the fitness landscape of each population is continuously changed by the competing population. Given the relative lack of empirical evidence for the importance of the Red Queen effect on biological evolution, Artificial Life techniques seem well-suited to study this penomenon [CM95]. From a computational perspective, competing co-evolutionary systems are appealing because the ever-changing fitness landscape, caused by the struggle of each species to take profit of the competitors' weaknesses, could be potentially exploited to prevent stagnation in local maxima.

Cliff and Miller realised the potentiality of co-evolution of pursuit-evasion tactics in evolutionary robotics. In the first of a series of papers [MC94], they provided an extensive review of the literature in biology and in differential game theory and introduced their 2D simulation of simple robots with "eyes". Later, they proposed a new set of performance and genetic measures in order to describe evolutionary progress which could not be otherwise tracked down due to the Red Queen effect [CM95]. Recently, they described some of the results where simulated robots with evolved eye-morphologies could either evade or pursue their competitors of several generations earlier and proposed some applications of the approach in biology and in the entertainment industry [CM96].

Despite the promising achievements described above, if one carefully looks at the

[2] The Red Queen is a figure, invented by novelist Lewis Carroll, who was always running without making any advancement because the landscape was moving with her.

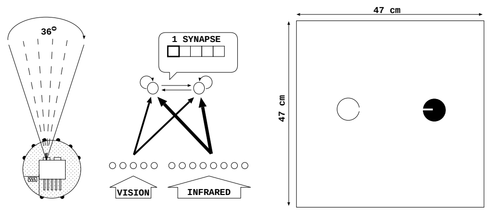

Figure 1.11 Left: Details of simulation of vision, of neural network architecture, and of genetic encoding. The prey differs from the predator in that it does not have 5 input units for vision. Each synapse in the network is coded by five bits, the first bit determining the sign of the synapse and the remaining four its strength. Right: Initial starting position for Prey (left, empty disk with small opening corresponding to frontal direction) and Predator (right, black disk with line corresponding to frontal direction) in the arena. For each competition, the initial orientation is random.

results described in the literature focusing on competitive co-evolution of pursuit-evasion behaviors, it is easy to notice that co-evolutionary benefits often come at the cost of several thousand individuals per population [Rey94], several hundred generations [CM96], or repeated trials of evolutionary runs with alternating success [Sim94]. Moreover, since all the experiments have been conducted in simulation, often the results cannot be directly applied to real robots, either because agent descriptions are too abstract or technically unfeasible, or because the fitness function takes into account global information (such as the distance between the competing agents). All these facts seem to greatly limit any prospect of exploiting the Red Queen effect for evolution of robotic controllers in the real world and for engineering purposes. The focus of the experiment here described is an investigation of the feasibility of this approach in more realistic conditions for evolutionary robotics.

The main goal of the experiments described here consists in studying the feasibility of co-evolutionary pursuit-evasion for evolving useful neurocontrollers for two Khepera robots in a simple but realistic scenario. We decided to study pursuit-evasion as a metaphor for predator-prey, this being a quite common and suggestive situation in nature. As often happens, predators and prey belong to different species with different sensory and motor characteristics. Thus, we employed two Khepera robots, one of which (the *Predator*) was equipped with a vision module while the other (the *Prey*) had a maximum available speed set to twice that of the predator (figure 1.10). Both individuals were also provided with eight infrared proximity sensors (six on the front side and two on the back) which had a maximum detection range of 3 cm in our environment. The two species would evolve in a square arena of size 47 x 47 cm with high white walls so that the predator could always see the prey (if within the visual angle) as a black spot on a white background.

Running co-evolutionary experiments with two or more robots within the same environment causes problems with the cables that connect the robots to the workstation for power supply and information exchange. Therefore, we decided to resort to a particular type of simulation extensively tested on Khepera which consists in sampling sensor activity at different distances and angles of the robot from the objects of the world (see [MLN96] for details). We have sampled infrared sensor activity of each robot in front of a wall and in front of another robot. These values were then separately stored away and accessed through a look-up table depending on the faced object. Simulation of the visual input required different considerations. The vision module K213 of Khepera is an additional turret which can be plugged-in directly on top of the basic platform. It consists of a 1D-array of 64 photoreceptors which provide a linear image composed of 64 pixels of 256 gray-levels each, subtending a view-angle of 36°. The optics are designed to bring into focus objects situated at distances between 5cm and 50cm while an additional sensor of light intensity automatically adapts the scanning speed of the chip to keep the image stable and exploit at best the sensitivity of receptors under a large variety of illumination intensities. The K213 vision turret incorporates a private 68HC11 processor which is used for optional low-level processing of the scanned image before passing it to the robot controller. One of these options is detection of the position in the image corresponding to the pixel with minimal intensity (in this case, only one byte of information is transmitted). Therefore, instead of simulating the response of the 1D-array of receptors resorting to complex and time-consuming ray-tracing techniques, we exploited the built-in facility for position detection of the pixel with minimal intensity and divided the visual angle in five sectors corresponding to five simulated photoreceptors (figure 1.11). If the pixel with minimal intensity was within the first sector, then the first simulated photoreceptor would become active; if the pixel was within the second sector, then the second photoreceptor would become active; etc. We made sure in a set of preliminary measurements that this type of input reduction was largely sufficient to reliably capture and represent all the relevant visual information available to the predator.

The robot controllers had the same architecture used for the experiment described in section 2, but the predator had 5 more input units corresponding to the visual module. Two populations of 100 individuals each were co-evolved for 100 generations. Each individual was tested against the best competitors of the ten previous generations (a similar procedure was used in [Sim94, Rey94, CM95]) in order to improve co-evolutionary stability. For each competition, the prey and predator were always positioned on a horizontal line in the middle of the environment at a distance corresponding to half the environment width (figure 1.11), but always at a new random orientation. The competition ended either when the predator touched the prey or after 500 motor updates (corresponding to 50 seconds at maximum on the physical robot). The fitness function Φ_c for each competition c did not require any sensor or motor measurement, nor any global position measure; it was simply *TimetoContact* normalized by the maximum number of motor updates TtC for the predator pr, and $1 - TtC$ for the prey py, further averaged over the number of competitions. Therefore the fitness values were always between 0 and 1, where 0 means worst.

Six evolutionary runs were performed, each lasting 100 generations. In all cases, after a few generations both the predator and the prey increased their fitness value and, for the remaining generations, we observed a set of oscillations in counterphase where either the predator or the prey reported better performance over the competitor. When we looked at the behavior of the two robots, we observed spontaneous evolution of obstacle avoidance, visual tracking, object discrimination (prey vs. wall), following, and a variety of other temporary behaviors (that is, lasting only few generations) which were tuned to the competitor behavior.

However, the tight evolutionary dynamics between the two competitors implied

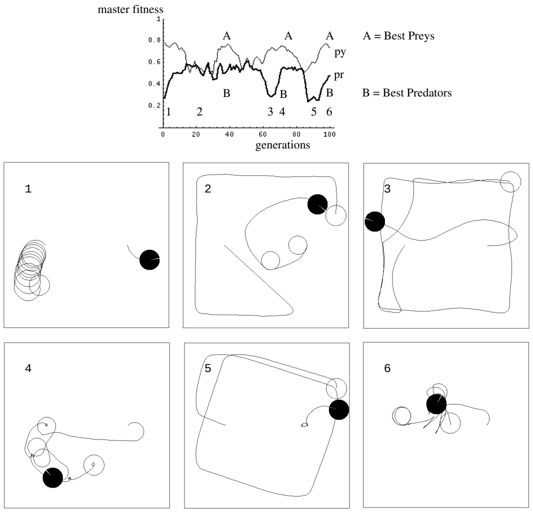

Figure 1.12 Top: Fitness of best individuals in Master Tournament. Letters indicate position of best prey and best predators. Numbers indicate position of individuals whose competitions are displayed below. Bottom: Behaviors recorded at interesting points of co-evolution, representing typical strategies. Black disk is predator, white is the prey (adapted from [FN97b]).

that the individuals of the last generation were not necessarily the best individuals of all generations, as it is usually the case in single-agent evolution. Rather, at each generation the best individuals are those individuals that report the best performance against the best competitors of the previous ten generations. The simplest way to discover the best predators and prey, is to organize a Master Tournament where each best individual is tested ten times against each best competitor of all generations. Top of Figure 1.12 shows the master fitness for each best individual across generations (fitness values are averaged over ten competitions and over hundred tournaments). A Master Tournament tells us two things: At which generation we can find the best prey and the best predator, and at which generation we are guaranteed to observe the most interesting tournaments. The first aspect is important for optimization purposes and applications, the latter for pure entertainment. The best individuals are those reporting the highest fitness when also the competitor reports the highest fitness (marked by letters A and B in the graph). Instead, the most entertaining tournaments are those that take place between individuals that report the same fitness level, because these are the situations where both species have the same level of ability to win over the competitor.

In the lower part of figure 1.12, behaviors of best competitors at critical stages of co-evolution, as indicated by Master Tournament data, give a more intuitive idea of how pursuit-evasion strategies are co-evolved. Initially, the predator tends to stop in front of walls while the prey moves in circles (box 1). Later, the prey moves fast at straight trajectories avoiding walls while the predator tracks it from the center and quickly attacks when the prey is closer (box 2). Interestingly, predators develop the ability to know how distant the prey are by using information on how fast their target moves in the visual field. Decrement of predator performance around generation 65 is due to a temporary loss of the ability to discriminate between walls and prey. As shown in box 3, the predator intercepts the prey, but it misses it crashing against the wall. Around generation 75, we have a typical example of the best prey (box 4); it moves in circles and, when the predator gets closer, it rapidly avoids it. This is quite interesting. Indeed, prey that move too fast around the environment sometimes cannot avoid an approaching predator because they detect it too late (IR sensors have lower sensitivity for a small cylindrical object, like another robot, than for a white flat wall). Therefore, it pays off to wait for the slower predator and accurately avoid it. However, some predators become smart enough to perform a small circle once they have missed the target, and re-attack until, by chance, the prey displays a side without IR sensors. As soon as the prey begin again moving around the environment, the predator develops a "spider strategy" (box 5): it slowly backs until it finds a wall where it waits for the fast-approaching prey. However, this strategy does not pay off when the prey stay in the same place. Finally, at generation 99 we have a new interesting strategy (box 6): the predator quickly tracks and reaches the prey which quietly rotates in small circles. As soon as the prey senses the predator, it backs and then approaches the predator (without touching it) on the side where it cannot be seen; consequently, the predator quickly turns in the attempt to visualize the prey which rotates around it, producing an entertaining dance.

These experiments have shown that competitive co-evolution is a promising technique for automatic evolution of complex behaviors without much effort in fitness design (see also [FN97a]). We have now built a special set of rotating contacts and sensors to detect robot contact that will allow us to run such experiments on the physical robots exploiting real vision. This setup permits to attach texture on the walls of the arena which could be used by the prey for camouflage and should prompt the predator to evolve efficient visual recognition abilities.

1.7 Evolvable Hardware

All the experiments described above were concerned with evolution of a control system implemented as a simulated neural network. However, it is interesting to investigate the possibility of evolving the robot hardware itself. A robot hardware is essentially composed of two things: the control circuit and the robot platform. There are *at least* two reasons why one might want to evolve hardware. On the one hand, evolving the physical circuit itself might result in much more efficient controllers in terms of the amount of physical resources employed and in terms of the *physical properties* of the circuit itself. As a matter of fact, software programs do not operate at the low level, but rather rely on a predesigned clock, pre-defined modularity of the hardware itself, and avoidance of all possible dynamical properties of the electronic components on the circuit. Such constraints are put there to facilitate logic design of programs, but are not necessary for producing input-output functions and might indeed be a sub-optimal utilization of the circuit itself. On the other hand, evolving *artificial brains* from scratch for complex and fixed robotic platforms is not necessarily the optimal way for developing a robot suited for a certain application. Both the robot morphology and the control system are intimately coupled and in nature both the nervous system and the body of organisms do co-evolve. At the same time, application specific robots

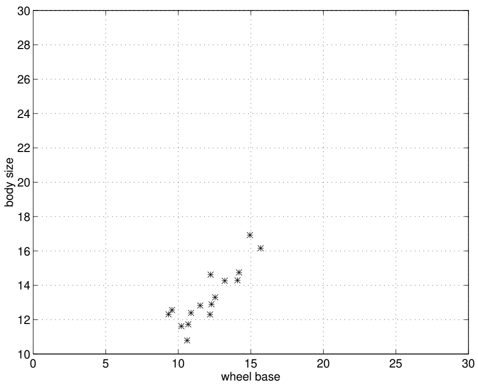

Figure 1.13 The relationship between evolved body size and wheel base for *low* sensor sensitivity on an obstacle avoidance task. Each data point corresponds to an evolved robot. Scales are in cm (reproduced from [LHL97] with author's permission).

should be designed so that they are suitable for the type of operations and interactions with the environment that are required by the task. A solution consists in co-evolving both the control system and some parameters of the paltform design for solving a particular task.

Adrian Thompson, working at the COGS group of the University of Sussex, has explored the possibility of evolving the control circuit of a mobile robot [Tho95, THH96] for an obstacle avoidance task. Silicon evolution is nowadays possible by using Field Programmable Gate Arrays, a recent generation of gate arrays which can be entirely reprogrammed via software [ST96]. A full hardware reconfiguration (modularity, logic operation, and connectivity) takes only a few milliseconds which makes it feasible to evolve genetically specified circuits. A cylindrical robot equipped with two wheels and two sonar sensors was put in a rectangular arena. In Thompson's experiments not only the circuit design was evolved, but also the global clock frequency and whether the signals flowing through were synchronized with the evolved clock or not. This means that the circuit properties now are tightly coupled with the temporal dynamics of the sonar sensors and of the robot's interactions with the environment. It took 35 generations to evolve a circuit that displayed the desired behavior; such circuit appropriately transformed the sensors signals in motor commands using 32 bits of RAM and 3 flip-flops.

Henrik Lund, working at the Artificial Intelligence Department of the University of Edinburgh, instead explored the possibility of evolving what he calls *body plans* together with the control system of such robots [LHL97]. A robot body plan is a specification of some parameters of the robotic platform; e.g., number, type, and position of sensors, body size, wheel size, etc. For a specific task only a subset of values and combinations of such parameter is well-suited. Inspired by biological evidence that body plans of all organisms are controlled by the same kind of genes [Day95], Lund evolved artificial chromosomes composed of a tree structure and of a list of real numbers. The tree structure encoded the control system and the sensor sensitivity and position, the list structure encoded the robot body plan, that is body size, wheel base, wheel radius, and motor time constants. He explored this approach

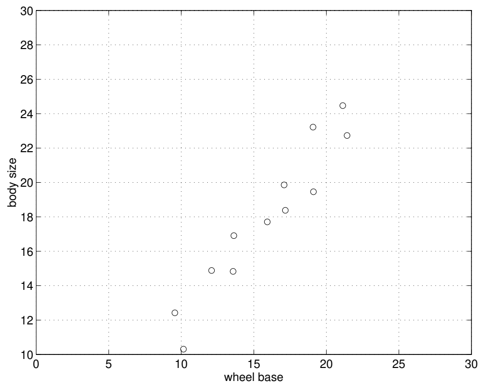

Figure 1.14 The relationship between evolved body size and wheel base for *high* sensor sensitivity on an obstacle avoidance task. Each data point corresponds to an evolved robot. Scales are in cm (reproduced from [LHL97] with author's permission).

by using a careful simulation of the robot and a simple task like obstacle avoidance. The results showed that, given a certain sensor sensitivity, the robots evolved for obstacle avoidance displayed a linear relationship between wheel base and body size (figure 1.13). Furthermore, for increased sensor sensitivity the linear relationship was maintained, but the body size and the wheel base became wider (figure 1.14). These results are explained by the fact that if obstacles can be perceived from distance, the robot has more time to react by turning and therefore a larger body and wheel size are suitable, whereas if obstacles can be sensed only when they are very close the robot must be capable of turning faster (smaller body and wheel base).

1.8 Conclusion

Evolutionary computation is a very powerful, yet general, method for developing autonomous mobile robots. *Evolutionary Robotics* is a young field of research. As such, it is still proceeding through an initial exploration phase, but the results described in this chapter indicate that it is evolving quickly and widely. The main difference between the evolutionary approach and human design (including several hand-crafted learning architectures) is that in the former case the robot autonomously develops its abilities and characteristics while it operates in the environment.

Nevertheless, there still are several challenges ahead [MC96]. One of these is the amount of time necessary to evolve robots that display the desired characteristics. For example, the experiment described in section 1.3 lasted approximately ten days (during which the Khepera robot continuously evolved day and night).This is a serious drawback for power supply. Although we can provide power for the Khepera robot through a serial cable and rotating contacts, not the same can be done with other bigger robots. Furthermore, in some real-world applications one might want to adapt the robot without a cable connection and in much shorter time.

I see two possible solutions –which can be also combined– to this problem. The first consists in evolving learning controllers. In other words, artificial evolution adapts high-level specifications of a neural controller that quickly learns using fast synaptic change. Initial results in this direction have already shown that the combination of evolution and learning provides faster adaptation and better solutions [NEP94, FM96b, NP96] for autonomous mobile robots. Other authors have also provided theoretical and experimental evidence for the synergetic effects of learning and evolution in artificial and biological adaptive systems [HN87, BM96]. The other solution consists in using incremental evolution. Instead of starting each new experiment from a population of random chromosomes, one can continue evolution from populations already evolved for simpler problems. At our lab, Francesco Mondada has had preliminary successful results in transferring controllers evolved for the Khepera robot on a much larger robot with six wheels, a different geometrical layout and sensors, by continuing evolution only for a few generations. Similarly, the last population evolved for the homing behavior described in section 1.3 was quickly adapted to new environments where the battery charger was positioned at different locations with respect to the light source [Flo96]. Also the experiments by Harvey *et al.* described in section 1.5 make use of incremental evolution. Harvey claims that incremental evolution is better suited for genetically converged populations and evolution is is treated like exploration, rather than search, driven mainly by mutation [Har92, Har93]. These issues are also directly related to the question of genetic encoding, that is what are the most suitable building blocks and encoding techniques for optimally exploiting the power of artificial evolution in a continuously evolving system facing different environments, new challenges, and modifiable hardware.

Acknowledgements

The experiments described in sections 1.2 and 1.3 were done in collaboration with Francesco Mondada, those described in section 1.6 in collaboration with Stefano Nolfi. I also wish to thank Henrik Lund for providing a copy of the original figures reproduced in section 1.7.

REFERENCES

[AL92] Ackley D. H. and Littman M. L. (1992) Interactions between learning and evolution. In Langton C., Farmer J., Rasmussen S., and Taylor C. (eds) *Artificial Life II: Proceedings Volume of Santa Fe Conference*, volume XI. Addison Wesley: series of the Santa Fe Institute Studies in the Sciences of Complexities, Redwood City, CA.

[BM96] Belew R. K. and Mitchell M. (eds) (1996) *Adaptive Individuals in Evolving Populations. Models and Algorithms.* Addison-Wesley, Redwood City, CA.

[Bro90] Brooks R. A. (1990) Elephants don't play chess. *Robotics and Autonomous Systems* 6: 3–15.

[Bro91] Brooks R. A. (1991) Intelligence without representation. *Artificial Intelligence* 47: 139–59.

[Bro92] Brooks R. A. (1992) Artificial Life and real robots. In Varela F. J. and Bourgine P. (eds) *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life.* The MIT Press/Bradford Books, Cambridge, MA.

[BSA83] Barto A. G., Sutton R. S., and Anderson C. W. (1983) Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* 13(5): 835–846.

[CDB96] Colombetti M., Dorigo M., and Borghi G. (1996) Behavior Analysis and

Training: A Methodology for Behavior Engineering. *IEEE Transactions on Systems, Man and Cybernetics-Part B* 26: 365–380.

[CHH93] Cliff D., Harvey I., and Husbands P. (1993) Explorations in evolutionary robotics. *Adaptive Behavior* 2: 73–110.

[CM95] Cliff D. and Miller G. F. (1995) Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In Morán F., Moreno A., Merelo J. J., and Chacón P. (eds) *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, pages 200–218. Springer Verlag, Berlin.

[CM96] Cliff D. and Miller G. F. (1996) Co-evolution of Pursuit and Evasion II: Simulation Methods and Results. In Maes P., Mataric M., Meyer J., Pollack J., Roitblat H., and Wilson S. (eds) *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA.

[Day95] Day S. (1995) Invasion of shapechangers. *New Scientist* 2001: 30–35.

[DS93] Dorigo M. and Schnepf U. (1993) Genetic-based machine learning and behavior based robotics: a new synthesis. *IEEE Transactions on Systems, Man and Cybernetics* 23: 141–154.

[Flo96] Floreano D. (1996) Evolutionary re-adaptation of neurocontrollers in changing environments. In Sincak P. (ed) *Proceedings of the Conference Intelligent Technologies*, volume II, pages 9–20. Efa Press, Kosice, Slovakia.

[FM94] Floreano D. and Mondada F. (1994) Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In Cliff D., Husbands P., Meyer J., and Wilson S. W. (eds) *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 402–410. MIT Press-Bradford Books, Cambridge, MA.

[FM96a] Floreano D. and Mondada F. (1996) Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B* 26: 396–407.

[FM96b] Floreano D. and Mondada F. (1996) Evolution of plastic neurocontrollers for situated agents. In Maes P., Mataric M., Meyer J., Pollack J., Roitblat H., and Wilson S. (eds) *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 402–410. MIT Press-Bradford Books, Cambridge, MA.

[FN97a] Floreano D. and Nolfi S. (1997) Adaptive behavior in competing co-evolving species. In Husbands P. and Harvey I. (eds) *Proceedings of the 4th European Conference on Artificial Life*. MIT Press, Cambridge, MA.

[FN97b] Floreano D. and Nolfi S. (1997) God save the red queen! competition in co-evolutionary robotics. In Koza J., Deb K., Dorigo M., Fogel D., Garzon M., Iba H., and Riolo R. L. (eds) *Proceedings of the 2nd International Conference on Genetic Programming*. Morgan Kaufmann, San Mateo, CA.

[Gib79] Gibson J. J. (1979) *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston.

[Gol89] Goldberg D. E. (1989) *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Redwood City, CA.

[Har92] Harvey I. (1992) Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In Varela F. J. and Bourgine P. (eds) *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press-Bradford Books, Cambridge, MA.

[Har93] Harvey I. (1993) Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In Langton C. (ed) *Artificial Life III*, pages 299–326. Addison Wesley, Redwood City, CA.

[HHC94] Harvey I., Husbands P., and Cliff D. (1994) Seeing The Light: Artificial Evolution, Real Vision. In Cliff D., Husbands P., Meyer J., and Wilson S. W. (eds)

From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior. MIT Press-Bradford Books, Cambridge, MA.

[HN87] Hinton G. E. and Nowlan S. J. (1987) How learning can guide evolution. *Complex Systems* 1: 495–502.

[KJ92] Kauffman S. A. and Johnsen S. (1992) Co-evolution to the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches. In Langton C., Farmer J., Rasmussen S., and Taylor C. (eds) *Artificial Life II: Proceedings Volume of Santa Fe Conference*, volume XI. Addison Wesley: series of the Santa Fe Institute Studies in the Sciences of Complexities, Redwood City, CA.

[LHL97] Lund H. H., Hallam J., and Lee W.-P. (1997) Evolving robot morphology. In *Proceedings of the IEEE 4th International Conference on Evolutionary Computation*. IEEE Press.

[Mae92] Maes P. (1992) Learning behavior networks from experience. In Varela F. J. and Bourgine P. (eds) *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. MIT Press-Bradford Books, Cambridge, MA.

[Mae93] Maes P. (1993) Behavior-based artificial intelligence. In Meyer J., Roitblat H. L., and Wilson S. W. (eds) *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA.

[Mar82] Marr D. (1982) *Vision*. Freeman, New York.

[MC92] Mahadevan S. and Connell J. (1992) Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence* 55(2): 311–365.

[MC94] Miller G. F. and Cliff D. (1994) Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In Cliff D., Husbands P., Meyer J., and Wilson S. W. (eds) *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA.

[MC96] Mataric M. and Cliff D. (1996) Challenges in Evolving Controllers for Physical Robots. *Robotics and Autonomous Systems* 19(1): 67–83.

[MFI93] Mondada F., Franzi E., and Ienne P. (1993) Mobile robot miniaturization: A tool for investigation in control algorithms. In Yoshikawa T. and Miyazaki F. (eds) *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513. Springer Verlag, Tokyo.

[Mil96] Millan J. (1996) Rapid, Safe, and Incremental Learning of Navigation Strategies. *IEEE Transactions on Systems, Man and Cybernetics: Part B: Cybernetics* 26: 408–420.

[Mil97] Millan J. (1997) Incremental acquisition of local networks for the control of autonomous robots. In Gerstner W., Germond A., Hasler M., and Nicoud J. (eds) *Proceedings of the International Conference on Artificial Neural Networks*. Springer-Verlag, Berlin.

[MLN96] Miglino O., Lund H. H., and Nolfi S. (1996) Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life* 2: 417–434.

[NEP94] Nolfi S., Elman J. L., and Parisi D. (1994) Learning and evolution in neural networks. *Adaptive Behavior* 3: 5–28.

[Nol97] Nolfi S. (1997) Using emergent modularity to develop control system for mobile robots. *Adaptive Behavior* 5: in press.

[NP96] Nolfi S. and Parisi D. (1996) Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior* 5: 75–98.

[Rey94] Reynolds C. W. (1994) Competition, Coevolution and the Game of Tag. In Brooks R. and Maes P. (eds) *Proceedings of the Fourth Workshop on Artificial Life*, pages 59–69. MIT Press, Boston, MA.

[Sim94] Sims K. (1994) Evolving 3D Morphology and Behavior by Competition. In Brooks R. and Maes P. (eds) *Proceedings of the Fourth Workshop on Artificial Life*, pages 28–39. MIT Press, Boston, MA.
[ST96] Sanchez E. and Tomassini M. (eds) (1996) *Towards Evolvable Hardware.* Springer-Verlag Notes in Computer Science, Berlin.
[THH96] Thompson A., Harvey I., and Husbands P. (1996) Unconstrained evolution and hard consequences. In Sanchez E. and Tomassini M. (eds) *Towards Evolvable Hardware. The Evolutionary Engineering Approach.* Springer Verlag, Berlin.
[Tho95] Thompson A. (1995) Evolving electronic robot controllers that exploit hardware resources. In Moran F. *et al.*. (ed) *Advances in Artificial Life: Proc. of ECAL95.* Springer Verlag, Barcelona.