Hardware Solutions for Evolutionary Robotics

Dario Floreano[1] and Francesco Mondada[1,2]

[1] Laboratory of Microcomputing
Swiss Federal Institute of Technology, CH-1015 Lausanne
[2] K-Team SA, CH-1028 Préverenges, Switzerland

Abstract. Evolutionary robotics— as other adaptive methods, such as reinforcement learning and learning classifier systems—can take considerable time and resources which require a careful evaluation of the hardware tools and methodologies employed. We outline a set of hardware solutions and working methodologies that can be used for successfully implementing and extending the evolutionary approach to complex environments, robots, and real-world applications. The issues discussed include the integration of simulation and real robots, design issues of evolvable robots, hardware requirements for incremental evolution, and hardware and software tools for monitoring and analysis.

1 Introduction

Evolutionary techniques applied to robot control can generate efficient, smart, and creative solutions which match the constraints imposed by the environment and the selection criterion. The power, flexibility, and generality of artificial evolution has often been exploited both for finding engineering solutions to difficult control problems of autonomous robots and for gaining new insights into the biological mechanisms of adaptive behavior (see, e.g., [25, 5, 23, 29, 19, 3, 21, 14], for thematic collections of several relevant papers). However, evolutionary robotics— as other adaptive methods, such as reinforcement learning and learning classifier systems—can take considerable time and resources [24] which require a careful evaluation of the hardware tools and methodologies employed.

In this contribution, we shall outline a set of hardware methodologies jointly developed and tested at our laboratory and at K-Team SA for reliable and controlled evolutionary experiments with mobile robots. The design issues and considerations described below can be used as a guideline for assessing the requirements and feasibility of a planned experiment in evolutionary robotics, but can also be generalized to experiments involving other types of learning methods. Although most of the solutions proposed in this paper are applicable to mobile robots only, the underlying methodological issues can be extended also to other robotic platforms equipped with adaptive control systems.

The paper is articulated around four major issues. In the next section we shall discuss a way to combine the advantages of simulations with those of real robots, introducing the concept of miniature mobile robots. We shall then discuss basic design principles and methodologies of robots that can be used for artificial evolution. Capitalizing on these principles, we shall also discuss hardware solutions which can support incremental evolution, namely modularity and cross-platform compatibility. Finally, we shall address the issue of monitoring and analysis of evolved robotic controllers.

2 Between Simulation and Real World

Evolutionary robotics has its roots in simulations. The first explorations in evolution of autonomous sensorimotor agents date back to the end of the Eighties,[3] when all studies were still carried out in computer simulations. A few years later, the appearance of more robust, flexible, and user-friendly robots, and a general awareness of the limitations of simulation methods [2], created a strong motivation for the first physical implementations of evolutionary robots [9, 17].

Despite the importance of keeping in mind the hard constraints of operating with physical robots, simulations still play an important role in evolutionary robotics for at least two sets of reasons:

- **Why simulations: The practical reason**. Researchers who actively contribute to this field have very different backgrounds, including computer science, robotics, psychology, biology, and philosophy. Only few of them have the technical skills and/or local support to build a robot and program it. Most of the commercial platforms available on the market have been developed by robot experts for robot experts, resort to very specific programming languages, and thus are not accessible to outsiders. This situation is currently improving, but a majority of robots still needs specific know-how. Finally, in universities, where software writing is considered "without costs", a real robot seems often too expensive in comparison to simulations.
- **Why simulations: The strategic reason**. Evolution of complex behaviors from scratch on a physical robot, even where technically feasible, would require an adaptive time which is too long to be practically exploited for industrial applications. Simulations can be of great help when properly integrated with tests on the physical robot. Current simulative methods have progressed from unrealistic grid-worlds to new methodologies that guarantee an acceptable transfer to the target robot under well-understood constraints [31, 26, 20] and can speed up evolutionary search by several orders of magnitude. One can develop initial behavioral skills in simulation and then continue on the physical robot, or combine evaluations on simulated and physical robots, or even build computationally efficient simulations that allow a good transfer to the real robot. Furthermore, we think that simulation is an important first step within an incremental evolutionary approach, which will be discussed below in section 4.

The miniature mobile robot Khepera has been designed as a research tool to fill the gap and allow a smooth transition between simulations and the real

[3] But appeared in the scientific press only a couple of years later [6, 32, 8]

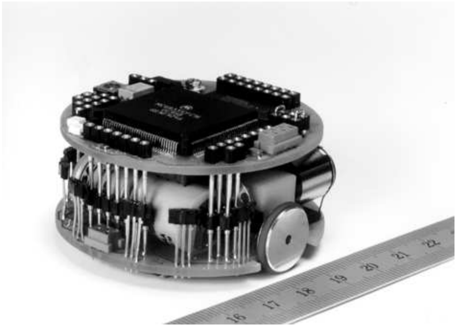

Fig. 1. The miniature mobile robot Khepera beside a ruler (in cm). Black pucks around the body are active infrared sensors. Rechargeable batteries are sandwiched between the motor base and the top structure; the latter hosts the microcontroller (black square), EPROM, RAM, and communication ports.

world. Initially conceived in 1991 at our laboratory by E. Franzi, A. Guignard, and F. Mondada, Khepera is currently employed in several hundred laboratories around the world for research in traditional and new-wave robotics.[4] If on the one hand its simplicity and interface make the Khepera as easy to work with as a simulation environment, on the other hand this robot cannot reach the full complexity of real-world applications. However, being a physical robot operating in a physical environment, it displays most of the characteristics of robots used for real-world applications. Therefore, it is well-suited for initial developments when one still wishes to run its own evolutionary models on the computer, but also wants to interface the program with a robot in a plug-and-play fashion in order to include real-world features in the evaluation of the genetic strings.

One of these features is *real-time operation*. In robot simulations, time is often a commodity that can be ignored or easily managed. All processes are synchronized, sub-routines are sequentially executed, the speed of sensorimotor loops is dictated by the number of intervening computations and by the power of the workstation CPU, and energy resources are not an issue. On the other hand, time constraints are ubiquitous in real robots and in biological organisms and, in the latter case, are often exploited by natural selection (for example, in several predator-prey scenarios). At the same time, several researchers have acknowledged the importance of evolving neural networks with temporal dynamics [1, 10, 18]. Khepera is a robot that can be attached to the serial port of a workstation and handled in the same way in which one would handle a simulator, but it also offers all time constraints of real-world robots. Depending on the working modality chosen by the user (see next section), the behavior of the robot is affected by the time taken by routine computation, the length of

[4] Here we will focus mainly on general design principles suitable for evolutionary robotics; a technical description of the Khepera robot can be found elsewhere [28].

messages exchanged by individual sub-components, and the type of communication between the robot and the workstation (if any). Here evolution of neural controllers with time dynamics has two advantages. Neurocontrollers can either adapt their own internal dynamics to those imposed by the experimenter on the robot (such as fixed action duration), or can exploit asynchrony and physical time delays to better cope with the environment. Furthermore, if controllers are evolved on a real robot, there is no need to include extra components in the fitness function in order to penalize complex architectures: if fast-thinking is important, sleek architectures and simple smart mechanisms will have higher reproduction chances over architectures that require heavy computations and complex memory handling.

Another feature that Khepera and real-world robots share is sensor and motor imprecisions. This well-known difference between simulations and real robots is of great importance in evolutionary robotics because artificial evolution often generates controllers that rely on the sensorimotor characteristics of the agent [9] more than control systems designed according to human logic. Interfacing one's own evolutionary software with a real robot not only generates more robust controllers, but can also qualitatively affect the type of controllers which are evolved.

3 Evolvable Robots

Building mobile platforms suitable for intensive learning mechanisms, such as artificial evolution, requires specific solutions. Although most of those outlined below have been incorporated in various stages of the realization of the Khepera robot, they are general enough to be used as guidelines for building other platforms.

3.1 Miniaturization

During the initial generations of an evolutionary run, most of the individuals display behaviors which might damage the robot shell, such as collisions against obstacles, improper handling of manipulators, pushing walls at sustained speed, etc. Whereas large robots have little chance of remaining operative for more than a few generations under these conditions, fundamental laws of physics endow miniature robots with higher mechanical robustness. In order to intuitively understand this feature, compare a robot of 50 mm in diameter crashing against a wall at 50 mm/s to a robot of 1 m in diameter crashing against the same wall at 1 m/s. The miniature robot will resist the collision, the other robot will probably report serious damages. However, it is important to choose the appropriate level of miniaturization. Although 1 cubic inch autonomous mobile robots are technically feasible [30], it would be advisable to choose a level of miniaturization where there is still a sufficient flexibility at the level of the development tools available, the components used, the performances achieved and the effort required by the development. Currently, the minimum attainable size for an autonomous learning robot which can be assembled with off-the-shelf components is around 2-3 cubic inches.

3.2 Interface granularity

Artificial evolution should be allowed to access sensors, motors, and other features of the robot at several levels, including the lowest-possible level. By restricting user's access to high-level commands and pre-processed sensory information, one certainly reduces the degrees of freedom of the controller, but might also hamper the adaptive power of the evolved solution. Evolution should be allowed to manipulate and combine fine details of the sensorimotor interface in the the most suitable way for the characteristics of the environment where the robot operates.

Think, for example, about the speed of the wheels. Some of the possible command levels are: go forward (speed and action duration are fixed); set speeds of wheels (action duration is fixed); set speeds and duration. One can go even further down, by-passing any PID algorithm to control motor speed. High-level commands are better suited for well-known and noise-free environments, whereas low-level commands give more adaptive power in unknown situations with variable levels of noise. These are typical situations where one wishes to use artificial evolution, rather than pre-assembling a set of macro-behaviors.

A similar reasoning goes for the sensory input. Pre-processed information is an efficient solution if one knows the features of the environment where the robot will operate. In other cases, raw sensory data, opportunistically exploited by evolutionary mechanisms, are likely to generate more robust controllers. For example, the vision module K213 [27] for the Khepera, a linear array of 64 photoreceptors spanning a 36° visual field, offers both low-level and high-level access to the data. Low-level access provides the gray-level image refreshed at a variable rate, whereas high-level functionalities allow on-board extraction of the pixel with the minimal or maximal activity, reading of only a part of the image, trasmission of thresholded values, etc. Clearly, pre-processed information might reduce the evolutionary search space and amount of computation required by the evolutionary controller, but it is necessary to ponder accurately whether this might *a priori* exclude the emergence of interesting solutions.

In contrast to robots employed for pre-programmed actions in well-known environments, the sensorimotor interface of evolvable robots should be organized in layers offering different levels of granularity, all directly accessible in parallel by the evolutionary engine.

3.3 Interface quality

Scientists and engineers develop over the years a preference for a set of standard computing and visualization tools. Just as biologists prefer to give public presentations with slides whereas computer scientists rather prefer transparencies, different research communities have different ways of working with a computer.

Getting used to a new software or to a new machine is often perceived as a nuisance and a waste of time.

Robots developed for an interdisciplinary research enterprise, such as artificial evolution, should be easily interfaceable to different platforms, include libraries for major language compilers, and be easily integrated with popular scientific packages. Plug-and-play feel is very important in this cross-disciplinary field because it encourages an increasing number of people to move from simulations to real robots. Flexibility, transportability, and an open architecture are other basic qualities of the interface of a robot intended for several purposes and user types.

3.4 Computation management

Artificial evolution requires three basic features: sustained electrical power for long periods, data storage for analysis, and some computational power. In order to meet these requirements, which are also shared by other learning mechanisms, the Khepera robot has been designed so that it can be attached to any computer through a serial connection and rotating contacts (figure 2). The serial connection provides electrical power and supports fast data communication (up to 57kBaud); it is simpler to implement and more reliable than infrared and/or radio links. This setup allows the user to run the evolutionary algorithm and all the control routines on the computer CPU which reads sensor activity and sends motor commands several times a second through the cable. All data concerning several generations can be conveniently stored on the computer hard disk for later analysis.

Real-time sensorimotor communication between the robot and the workstation, however, shows its limitations when the amount of data to be transferred is relevant, such as with vision, and when the environment requires fast responses (dynamic environments). Furthermore, the user might wish to test or further train an evolved controller on large environments without a cable connection. Autonomous robots thus must have more computational power and memory resources on-board than other types of robots, such as tele-operated robots or pre-programmed machines. If sufficient memory and computational power is available on-board, the evolved control system can be cross-compiled for the processor on the robot and downloaded.

Alternatively, one can run the control system on the robot itself and the evolutionary algorithm on the computer. This latter hybrid solution requires only one data transmission per individual for sending the artificial chromosome from the computer to the robot (and optionally reading the fitness value returned by the previously tested individual). We have used this method for predator-prey co-evolutionary experiments with two Khepera robots, where any delay in visual processing and/or motor reaction due to transmission time would be readily exploited by the opponent [11].

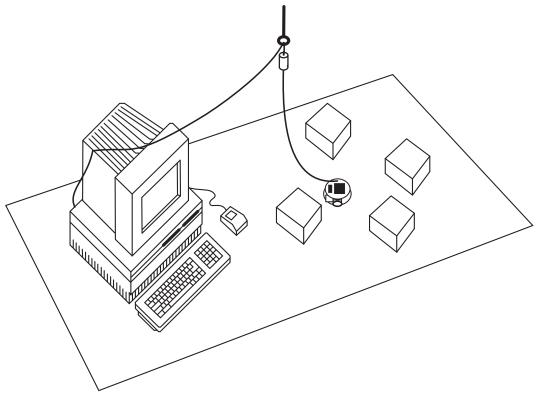

Fig. 2. The robot communicates with a computer via a thin suspended serial cable which also supplies electrical power. Rotating contacts prevent cable twisting and ensure noise-free communication. This solution can be used to control the robot from the computer, download the evolved controllers on the robot processor, or use a hybrid approach (see text).

4 Incremental Evolution

The future and applicability of evolutionary robotics heavily relies on methods of incremental evolution. Incremental evolution consists in gradually evolving a controller on a series of tasks of increasing complexity. There are several reasons why an incremental approach is important in evolutionary robotics.

Attempting to evolve a complex task from a limited collection of randomly created strings is difficult because none of the individuals in the initial generation might display competences which can be credited by the fitness function. Various approaches have been suggested, such as gradually increasing the complexity of the environment and/or modifying the fitness function during evolution [7, 17, 13]. In some circumstances, changing the environment is equivalent to modifying the morphology of the robot. For example, instead of attempting to evolve from scratch a complex behavior based on vision and appropriate control of a gripper module, it can be more fruitful to proceed gradually from a stripped-down version of the robot and add new components at later stages [12]. Incremental solutions that can cope with extendable hardware architectures are well-suited for open-ended evolution where the task under consideration can change over time in unpredictable ways [16] depending on current requirements.

Another reason for pursuing an incremental approach is scalability to robots suitable for real-world applications. Such robots are usually larger and more fragile than miniature robots suitable for research. We think that a viable solution consists in incrementally evolving behavioral competences starting from simulations, then gradually move to simple robots, and eventually continue on

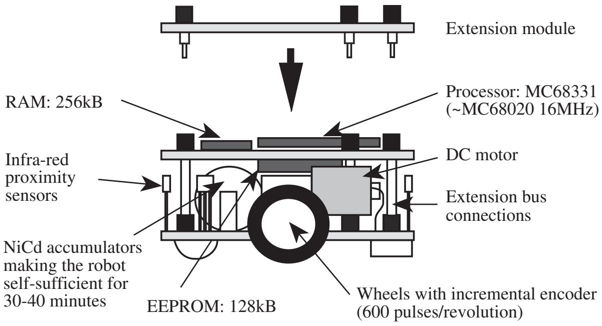

Fig. 3. Khepera robot structure and extension possibilities.

more complex robots. Cross-platform evolution is a form of incremental evolution where previously evolved blocks are gradually adapted, combined, and extended to accomodate new morphologies, sensorimotor interfaces, and changing task requirements.

Incremental evolution requires novel strategies capable of coping with variable fitness landscapes, variable-length genotypes, ontogenetic adaptation mechanisms, variable genotype-phenotype mappings, and modularity. Interesting methods tackling some of these issues have already been suggested [15], but much work remains to be done in the years ahead. In the next two sections we will address some hardware solutions that can support investigations in incremental evolution, namely modularity and cross-platform compatibility.

4.1 Modularity

Hardware modularity enables different possible configurations and experiments using the same basic components. It also means the possibility of adding extensions and, globally, cheaper equipment. Software modularity means flexibility and possibilities for extensions as new modules modules become available on the robot.

For example, at the hardware level the Khepera robot (figure 3) has an extension bus that makes it possible to add turrets on the top of the basic configuration, depending on the needs of the experiments to be done. This modularity is based on a parallel and a serial bus. The parallel bus can be used for simple extensions directly under control of the main Khepera processor. The serial bus implements a local network for inter-processor communication. Using this second bus, other processors can be connected to the main one in order to build a multi-processor structure centred on the Khepera main processor. This kind of structure has the advantage that one can employ additional computational devices on extension modules, thus keeping the main processor free for global management of the robot behavior.

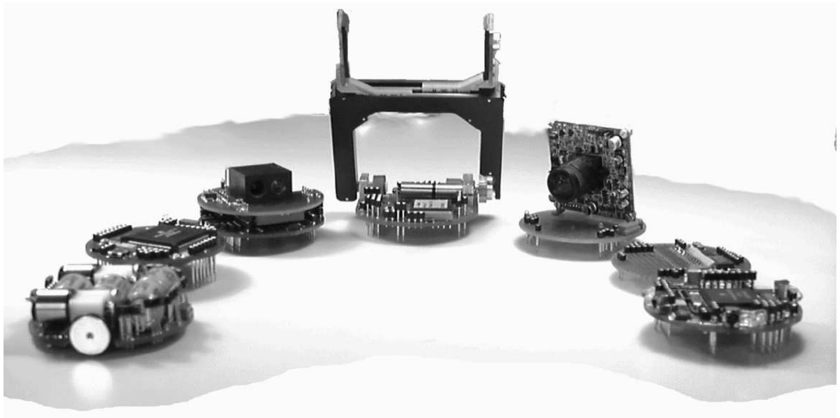

Fig. 4. Some modules of the Khepera robot. From left: basic platform with motors and batteries, microprocessor platform, linear vision, gripper, CCD camera, general input-output, and infrared communication.

Some of the hardware modules currently co-developed in the lab and at K-Team SA for the Khepera robot include a gripper with two degrees of freedom (elevation and grasping), a linear vision module (64 photoreceptors with automatic light sensitivity adjustement), a black-and-white or color CCD camera, a general input-output module where one can add several new sensors and actuators, a miniature GPS system for localisation, a radio link module, and an infrared communication module (figure 4).

At the software level, modularity is needed to support the multi-processor structure of the robot. It consists of a flexible protocol that recognizes all added extension modules when the robot is powered, informing the main processor about all functionalities available in each extension as well as the procedures for activating these functionalities. The BIOS of the Khepera, which includes all basic procedures for robot management, is also based on a modular structure. Motors, sensors, and timing functions are grouped into distinct modules to simplify management of the robot and improve software robustness. The main software also supports remote control and down-loading of specific applications through the serial cable. Such software structure simplifies the task of the user, who can easily add her own software to the management modules already implemented on Khepera.

4.2 Cross-platform compatibility

Cross-platform compatibility in mobile robotics is a very unusual feature, especially between robots of very different sizes. Software compatibility alone is not sufficient to guarantee a good transfer of algorithms between different robots.

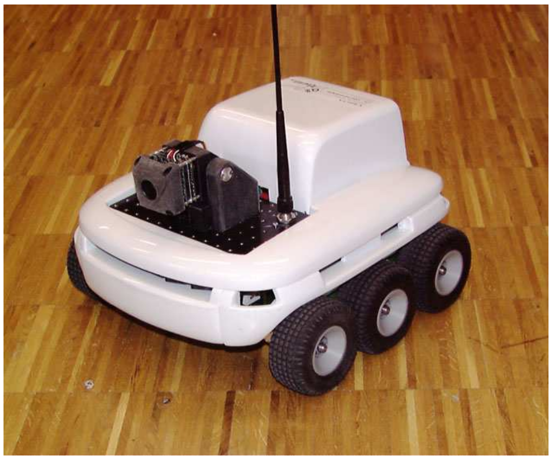

Fig. 5. The Koala robot, despite its look and size (31 x 32 x 18 cm), has operating features similar to those of the Khepera robot.

Mechanical structure, sensor caracteristics, and many other aspects of the physical platform also play an important role in the feasibility of the transfer.

Koala (figure 5) is a medium-size very performant mobile robot that has been designed to support transfers from the Khepera robot. Despite its better performance, size, shape, and look, the Koala robot is very similar to the Khepera in many essential aspects. The six wheels of Koala are driven by two motors, as for Khepera, each controlling one side of the platform. The central wheels on the two sides are lower than the others, so that rotation of the whole platform is very similar to that of the Khepera. The proximity sensors of the Koala are based on the same concept used for those of the Khepera, but the measurement range is scaled up to the larger size of the Koala. Also, the number of sensors has been changed from 8 on the Khepera to 16 on the Koala. The Khepera hardware modularity described above has also been both supported and scaled up in the Koala. In addition to the serial extension bus of the Khepera, the Koala is equipped with a fast inter-processor parallel communication bus to support larger amounts of data. At the software level, both robots are compatible, having the same low-level BIOS software and the same comunication facilities.

All these caracteristics allow a smooth transfer from the Khepera to the Koala, just like from simulations to the Khepera. The development and monitoring tools are the same for both robots and the code itself, even when downloaded (see section 3.4 above), is very similar. Preliminary screening of the evolving populations can start on simulations, than gradually move to the Khepera, and eventually continue on the Koala. In such a framework, the incremental approach suggested by Harvey [15] seems very interesting because, after an initial stage, evolving populations are almost always converged. That means that the

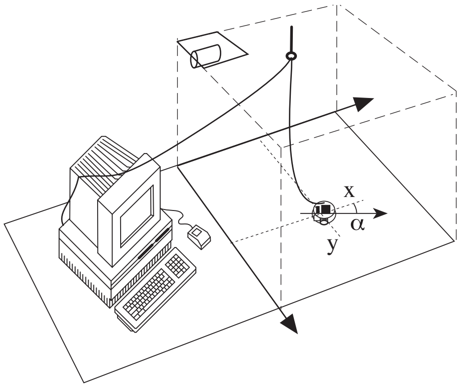

Fig. 6. The laser positioning device is composed of two elements; a portable laser emitting device on the ceiling of the room and an additional module plugged on the top of the robot which measures parameters of the laser beam and computes the exact robot position. This information is used only for neuroethological analysis.

number of maladaptive individuals tested on the physical robots is very limited with obvious benefits for the hardware platform.

5 Monitoring and Analysis

Artificial evolution requires tools for monitoring the evolutionary process and analysing the evolved controllers. In engineering applications monitoring and analysis are necessary for ensuring the stability of the learning process and of the evolved machine; in artificial life, they are required to understand the course of evolution and to establish relationships between evolved solutions and environmental constraints.

When artificial evolution is run in simulation, all the variables are readily available and analysis is technically straightforward. When it comes to evolving real robots, experimental results are often displayed as hand-drawn trajectories, photographs of the environment, or—at best—statistics on discrete events, such as number of collisions, of collected objects, etc. These methods are not sufficient to understand the tightly coupled dynamics between the evolved controllers and the environment. In contrast to disembodied machine learning systems (e.g., neural networks trained on pre-assembled data sets), evolutionary controllers develop mechanisms that depend on the type of interactions with the environment; the input distribution is determined both by the environment features and by the motor actions of the agent, which—in turn—depend on the sensory input [32]. This implies that one cannot hope to understand an evolved controller by isolating it from the environment and looking at its structure.

Understanding the course of artificial evolution and the functioning of evolved

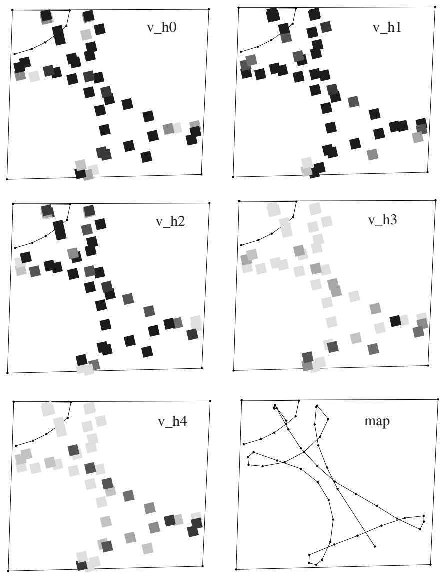

Fig. 7. Neural correlates of behavior. The evolved robot navigates in an environment and periodically returns to a recharging station. The controller is a neural network with five hidden neurons and fully recurrent connections. Each box plots the activity of one neuron every 100 ms while the evolved robot freely moves in the environment. Darker squares mean higher node activations. The robot starts in the lower portion of the arena. The recharging area is visible in the top left corner of each box. The bottom-right window plots only the trajectory. From [10]; ©1996 IEEE.

mechanisms requires a method to measure and correlate controller states with behavioral patterns. This implies two efforts. On the hardware side, the robotic setup must be designed so that physical displacements of the robot can be recorded on a fine time scale. On the software side, it should be possible to visualize, analyze, and correlate all the data (robot actions and controller states) at several levels. This procedure is similar to that employed in neuroethology where measurable sensory stimuli and motor actions are correlated with neural activity *in vivo* while the organism freely interacts with a controlled environment.

Data acquisition and analysis can be achieved in different ways. For example, a solution developed at the University of Edinburgh consists in placing LEDs on the robot and using a camera connected to a frame grabber which provides position data in real time [22]. However, this set up does not allow neuroethological analysis, unless also controller data are integrated with the position acquisition software. At our laboratory we have tested another approach, using the design principles described above (figure 6). A device emitting laser beams at predefined angles and frequencies was positioned in the environment and the Khepera was equipped with an additional turret capabable of detecting laser beams and computing in real-time the robot's displacements. This computation was carried out on a processor placed on the additional turret which did not interfere with the main controller on the robot. Every 100 ms robot position and controller variables where sent to a programmable acquisition software [4] which instantaneously processed and visualized data while the robot freely interacted with the environment (figure 7).

6 Conclusion

It has been claimed elsewhere that evolutionary robotics might be limited to simple experiments in highly controlled environments and might be applicable only to few robotic platforms [24]. In this paper we have outlined a set of hardware solutions and working methodologies that can be used for successfully extending the evolutionary approach to complex environments, robots, and real-world applications. The principles described in this paper are simple and general enough to be implemented on a variety of different robotic setups, and have all been exploited during various realization phases of the robots developed at our laboratory.

Artificial evolution remains a very powerful and general technique for automatically developing autonomous robots expected to operate in partially unknown environments and to investigate mechanisms of biological adaptive behavior. The time required to generate interesting solutions (which is comparable or shorter than the time required by other learning techniques, such as reinforcement learning) does not imply that the approach is not viable; it simply means that one should use it only for problems where traditional AI methods fail to produce interesting and useful controllers.

Acknowledgements

The authors wish to thank Edo Franzi and André Guignard for important contributions in the design of the hardware components described above. Khepera and Koala are trademarks of K-Team SA.

References

1. R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122, 1992.
2. R. A. Brooks. Artificial Life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*. The MIT Press/Bradford Books, Cambridge, MA, 1992.
3. R. A. Brooks and P. Maes, editors. *Artificial Life IV. Proceedings of the Fourth International Conference on Artificial Life*, Cambridge, MA, 1994. MIT Press.

4. Y. Cheneval. Packlib, an interactive environment to develop modular software for data processing. In J. Mira and F. Sandoval, editors, *From Natural to Artificial Neural Computation, IWANN-95*, pages 673–682, Malaga, 1995. Springer Verlag.
5. D. Cliff, P. Husbands, J. A. Meyer, and S. W. Wilson, editors. *From Animals to Animats: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1994. MIT Press/Bradford Books.
6. D. T. Cliff. Computational neuroethology: a provisional manifesto. In J. A. Meyer and S. W. Wilson, editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1991.
7. D. Floreano. Emergence of Home-Based Foraging Strategies in Ecosystems of Neural Networks. In J. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1993.
8. D. Floreano, O. Miglino, and D. Parisi. Emergent complex behaviours in ecosystems of neural networks. In E. Caianiello, editor, *Parallel Architectures and Neural Networks*. World Scientific Press, Singapore, 1991.
9. D. Floreano and F. Mondada. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *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, 1994.
10. D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:396–407, 1996.
11. D. Floreano, S. Nolfi, and F. Mondada. Co-evolutionary Robotics: From Theory to Practice. In preparation, 1997.
12. D. Floreano and J. I. Urzelai. Evolution and learning in autonomous robots. In D. Mange and M. Tomassini, editors, *Bio-Inspired Computing Systems*. PPUR, Lausanne, 1998.
13. F. Gomez and R. Miikkulainem. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.
14. T. Gomi, editor. *Evolutionary Robotics. From intelligent robots to artificial life.* AAI Books, Kanata, Canada, 1997.
15. I. Harvey. Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press-Bradford Books, Cambridge, MA, 1992.
16. I. Harvey. Artificial evolution for real problems. In T. Gomi, editor, *Evolutionary Robotics*, pages 187–220. AAI Books, Ontario, Canada, 1997.
17. I. Harvey, P. Husbands, and D. Cliff. Seeing The Light: Artificial Evolution, Real Vision. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1994.
18. I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary Robotics: The Sussex Approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.
19. P. Husbands and I. Harvey, editors. *Proceedings of the Fourth European Conference on Artificial Life*, Cambridge, MA, 1997. MIT Press/Bradford Books.

20. N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life*, Cambridge, MA, 1997. MIT Press.
21. C. G. Langton and K. Shimohara, editors. *Artificial Life V. Proceedings of the Fifth International Conference on Artificial Life*, Cambridge, MA, 1997. MIT Press.
22. H. H. Lund, E. de Ves Cuenca, and J. Hallam. A Simple Real-Time Mobile Robot Tracking System. Technical Report 41, Dept. of Artificial Intelligence, University of Edinburgh, 1996.
23. P. Maes, M. Matarić, J-A. Meyer, J. Pollack, H. Roitblat, and S. Wilson, editors. *From Animals to Animats: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1996. MIT Press/Bradford Books.
24. M. Matarić and D. Cliff. Challenges in Evolving Controllers for Physical Robots. *Robotics and Autonomous Systems*, 19(1):67–83, 1996.
25. J. A. Meyer, H. L. Roitblat, and S. W. Wilson, editors. *From Animals to Animats: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, 1993. MIT Press/Bradford Books.
26. O. Michel. *Khepera simulator Package.* LAMI, Swiss Federal Institute of Technology in Lausanne, Version 2.0 edition, 1996. Freeware mobile robot simulator downloadable from http://diwww.epfl.ch/lami/team/michel/khep-sim/.
27. F. Mondada and E. Franzi. *K213 Vision Turret User Manual.* K-Team S.A., 1028 Prévérenges, Switzerland, Version 1.0 edition, 1995.
28. F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513, Tokyo, 1993. Springer Verlag.
29. F. Morán, A. Moreno, J. J. Merelo, and Chacón P., editors. *Proceedings of the Third European Conference on Artificial Life*, Berlin, 1995. Springer-Verlag.
30. J-D. Nicoud and O. Matthey. Developing intelligent micro-mechanisms. In *Proceedings of the Conference on Micromechatronics and Human Science*, Nagoya, Japan, 1997. IEEE Press.
31. S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, pages 190–197, Boston, MA, 1994. MIT Press.
32. D. Parisi, F. Cecconi, and S. Nolfi. Econets: Neural networks that learn in an environment. *Network*, 1:149–168, 1990.

This article was processed using the LaTeX macro package with LLNCS style