Experimental Robotics III, Proceedings of the 3rd International Symposium on Experimental Robotics, Kyoto, Japan, October 28-30, 1993, Springer Verlag, London, 1994, pp501-513.

Mobile robot miniaturisation: A tool for investigation in control algorithms.

Francesco Mondada, Edoardo Franzi, and Paolo Ienne

Swiss Federal Institute of Technology
Microcomputing Laboratory
IN-F Ecublens, CH-1015 Lausanne
E-mail: Francesco.Mondada@di.epfl.ch
Edoardo.Franzi@di.epfl.ch
Paolo.Ienne@di.epfl.ch

Abstract

The interaction of an autonomous mobile robot with the real world critically depends on the robots morphology and on its environment. Building a model of these aspects is extremely complex, making simulation insufficient for accurate validation of control algorithms.

If simulation environments are often very efficient, the tools for experimenting with real robots are often inadequate. The traditional programming languages and tools seldom provide enought support for real-time experiments, thus hindering the understanding of the control algorithms and making the experimentation complex and time-consuming.

A miniature robot is presented: it has a cylindrical shape measuring 55 mm in diameter and 30 mm in height. Due to its small size, experiments can be performed quickly and cost-effectively in a small working area. Small peripherals can be designed and connected to the basic module and can take advantage of a versatile communication scheme. A serial-link is provided to run control algorithms on a workstation during debugging, thereby giving the user the opportunity of employing all available graphical tools. Once debugged, the algorithm can be downloaded to the robot and run on its own processor.

Experimentation with groups of robots is hardly possible with commercially available hardware. The size and the price of the described robot open the way to cost-effective investigations into collective behaviour. This aspect of research drives the design of the robot described in this paper. Experiments with some twenty units are planned for the near future.

1. Introduction

Today the mobile robotics field receives great attention. There is a wide range of industrial applications of autonomous mobile robots, including robots for automatic floor cleaning in buildings and factories, for mobile surveillance systems, for transporting parts in factories without the need for fixed installations, and for fruit collection and harvesting. These mobile robot applications are beyond the reach of current technology and show the inadequacy of traditional design methodologies. Several new control approaches have been attempted to improve robot interaction with the real world aimed at the autonomous achievement of tasks. An example is the *subsumption architecture* proposed by Brooks [1] which supports parallel processing and is modular as well as robust. This approach is one of the first solutions systematically implemented on real robots with success. Other researchers propose new computational approaches like *fuzzy logic* [2] or *artificial neural networks* [3].

The interest in mobile robots is not only directed toward industrial applications. Several biologists, psychologist and ethologists are interested in using mobile robots to validate control structures observed in the biological world. Franceschini [4] uses a robot to validate the structure of the retina observed on a fly, Beer [5] to replicate the mechanism that coordinates leg movements in walking insects, Deneubourg [6] to get a better understanding of collective behaviour in ant colonies.

All these research activities are based on mobile robot experimentation. A simpler way to validate control algorithms is to use simulations, but the simplifications involved are too important for the results to

be conclusive. The control algorithm embedded in the robot must consider its morphology and the properties of the environment in which it operates [7]. Real world features and anomalies are complex and difficult to modelise, implying that the experimentation of control algorithms through simulation can only be used in preliminary study but cannot prove the success of the control algorithm in the real world. The sole way to validate an algorithm to deal with these problems is to test it on a real robot [8].

Many robots have been designed to perform experiments on control algorithms but only a few make cost-efficient experiments possible. Brooks has designed several robots with effective electronics and mechanics [9]. The control algorithms are programmed in the subsumption behavioural language, taking into account software modularity, and real-time and parallel processing. Unfortunately, during experiments, only a few tools are available to improve the understanding of the control process. Moreover, the custom programming language makes code portability and algorithm diffusion difficult. Steels [10] uses a video-camera to record robot actions during experiments but all the data concerning the robot control process are available only at the end of the experiment. Other platforms, such as the Nomad robot [11], make real-time interaction possible via a radio link, and have standard programming languages, but the size of the robot and the environment it requires make experimentation uncomfortable.

The lack of a good experimentation mobile robot for single-robot experiments, means that it is impossible today to perform collective-behaviour experiments. The programming environment and the real-time visualisation tools are totally insufficient for this purpose.

The development of the miniature mobile robot *Khepera* addresses the problems mentioned above. Its hardware is designed so that it is small enough for the operation of several at the same time and in small areas, for example on a desk-top. Modularity allows new sensors and actuators to be easily designed and added to the basic structure. A versatile software structure is provided to help the user to debug the algorithms and to visualise the results.

2. Hardware

Miniaturisation is an important challenge for industry: CD players, computers, video cameras, watches and other consumer products need to implement many functionalities in a small volume. In the robotics field many applications need small actuators, small teleoperated machines or tiny autonomous robots. Dario [12] gives a comprehensive description of the research field and of the available technology. In the Khepera design, miniaturisation is the key factor in making cost-effective experimentations possible both for single or multiple robot configurations.

2.1. Generalities

The robot presented in this paper is only a first step in the direction of miniaturisation. Dario define this category of robots as *miniature robots*. They measure no more than a few cubic centimetres, generate forces comparable to those applied by human operators and incorporate conventional miniature components. The next miniaturisation step needs special fabrication technologies, today in development. Khepera uses electronic technology available today: the new family of 683xx microcontrollers from Motorola makes the design of complete 32 bit machines extremely compact. *Surface mounted devices* (SMD) allow an important increase in component density on printed circuit boards. New compact sensors, including some signal preprocessing on the sensing chip, reduce the need of additional circuitry. Only the mechanical parts (wheels, gears, manipulator) are built expressly for Khepera, as well as the magnetic sensors for counting the wheel revolutions.

The design of such miniaturised robots demands a great effort spanning several fields. The result is a complex compromise between functionalities to be implemented, available volume, current technology, power requirements, etc.

Khepera is composed of two main boards (figure 2).

Figure 1. The Khepera robot.

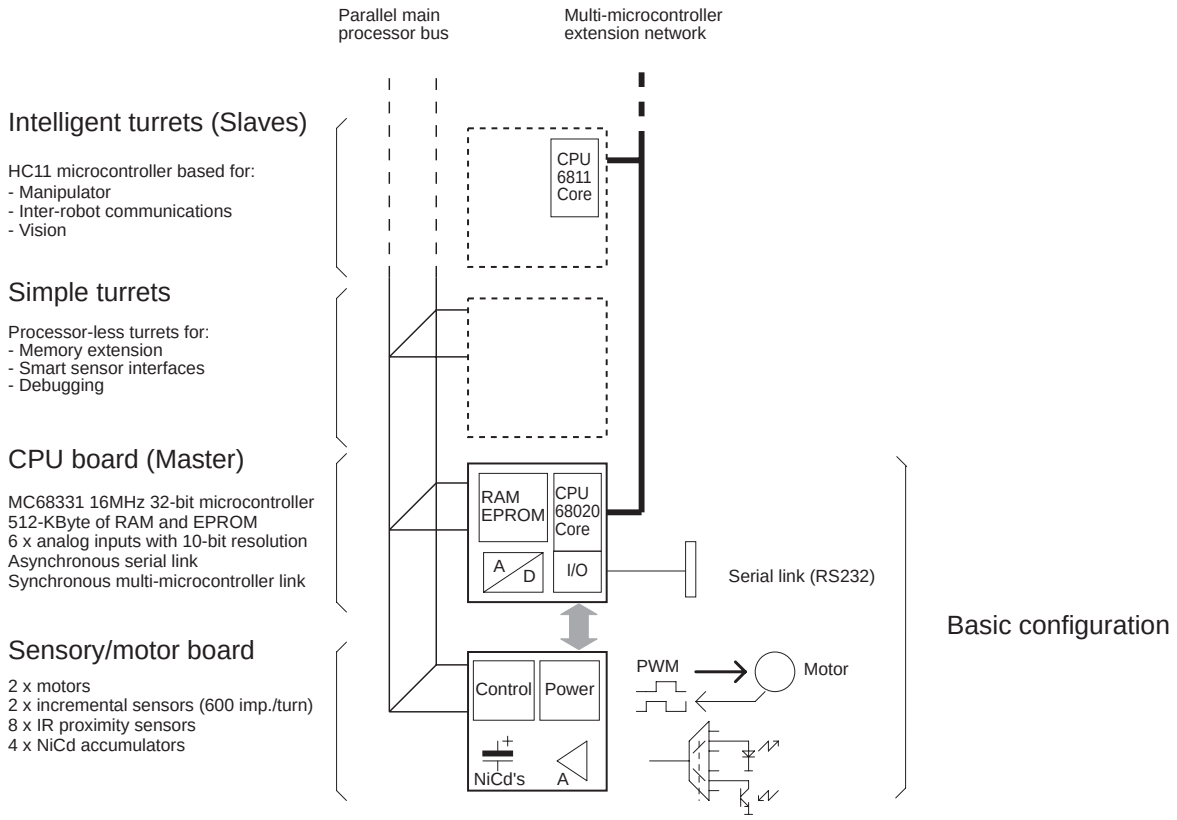

Figure 2. Khepera hardware architecture.

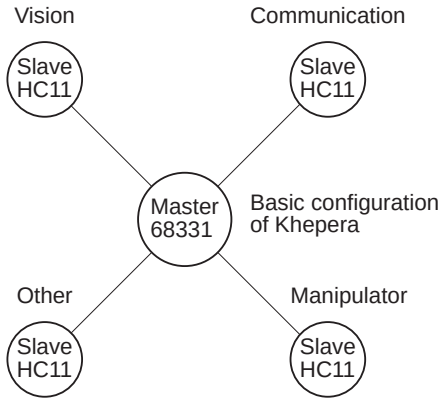

Figure 3. Khepera communication network topology.

Application-specific extension turrets for vision, for inter-robot communications, or which are equipped with manipulators can be directly controlled via the Khepera extension busses. Khepera can be powered by an external supply when connected for a long time to a visualisation software tool; however, on-board accumulators provide Khepera with thirty minutes of autonomous power supply.

2.2. Distributed processing

One of the most interesting features of Khepera is the possibility of connecting extensions on two different busses. One parallel bus is available to connect simple experimentation turrets. An alternative and more sophisticated interface scheme implements a small local communication network; this allows the connection of intelligent turrets (equipped with a local microcontroller) and the migration of conventional or neural pre-processing software layers closer to the sensors and actuators. This communication network (figure 3) uses a star topology; the main microcontroller of the robot acts as a master (at the centre of the star). All the intelligent turrets are considered as slaves (on the periphery of the star) and use the communication network only when requested by the master.

This topology makes it possible to implement distributed biological controls, such as arm movement coordination or feature extraction and pre-processing in the vision, as observed in a large number of insects. The multi-microcontroller approach allows the main

microcontroller of Khepera to execute only high level algorithms; therefore attaining a simpler programming paradigm.

2.3. Basic configuration

The new generation of Motorola microcontrollers and in particular the MC68331 makes it possible to build very powerful systems suitable for miniature neural control. Khepera takes advantage of all the microcontroller features to manage its vital functionality. The basic configuration of Khepera is composed of the CPU and of the sensory/motor boards.

The CPU board is a complete 32 bit machine including a 16 MHz microcontroller, system and user memory, analogue inputs, extension busses and a serial link allowing a connection to different host machines (terminals, visualisation software tools, etc.). The microcontroller includes all the features needed for easy interfacing with memories, with I/O ports and with external interruptions. Moreover, the large number of timers and their ability to work in association with the I/O ports indicate that this device is the most important component in the design.

The sensory/motor board includes two DC motors coupled with incremental sensors, eight analogue *infra-red* (IR) proximity sensors and on-board power supply. Each motor is powered by a 20 kHz *pulse width modulation* (PWM) signal coming from a dedicated unit of the microcontroller. These signals are boosted by complete four-quadrant NMOS H bridges. Incremental sensors are realised with magnetic sensors and provide quadrature signals with a resolution of 600 impulsions per wheel revolution. IR sensors are composed of an emitter and of an independent receiver. The dedicated electronic interface is built with multiplexers, sample/hold's and operational amplifiers. This allows the measurement of the absolute ambient light and the estimation, by reflection, of the relative position of an object from the robot.

2.4. Additional turrets

To make experiments involving environment recognition, object detection, object capture and object recognition possible, two intelligent turrets have been designed and built: one for stereoscopic vision, the other containing a manipulator.

The stereoscopic vision turret employs two 64 pixel linear photoelement arrays and a dedicated optical element. The analogue value of each pixel is coded on 16 grey levels. To obtain useable data under a wide spectrum of enlightenment conditions, an additional sensor is used to perform as an automatic iris: the integration time necessary for the photoelement arrays

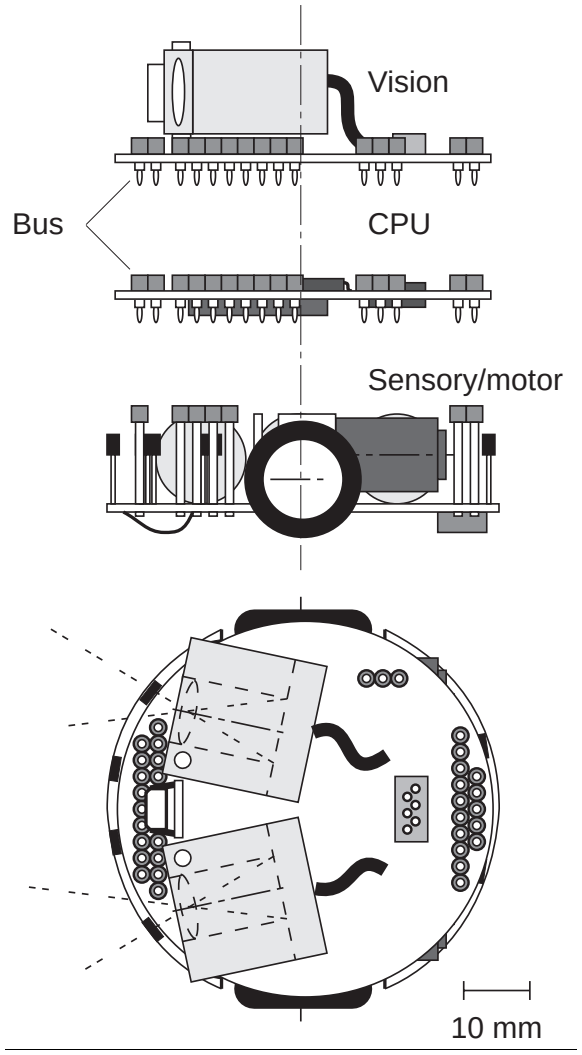

Figure 4. Khepera physical structure: Basic sensory/motor, CPU and vision boards.

is controlled by intensity of the ambient light. Mondada *et al.* [8] proved the validity of this stereoscopic vision in robot navigation (spatial frequency filtering was used in obstacle detection and avoidance).

The manipulator turret makes Khepera capable of an interaction with objects of its environment. Two DC motors control the movements of the manipulator (elevation and gripping). Different classes of objects can be detected by the gripper sensors which measure sizes and resistivities.

Robots displaying collective behaviour need means to perform inter-robot communications and localisation. Turrets providing Khepera with these functionalities are under study at the time of writing.

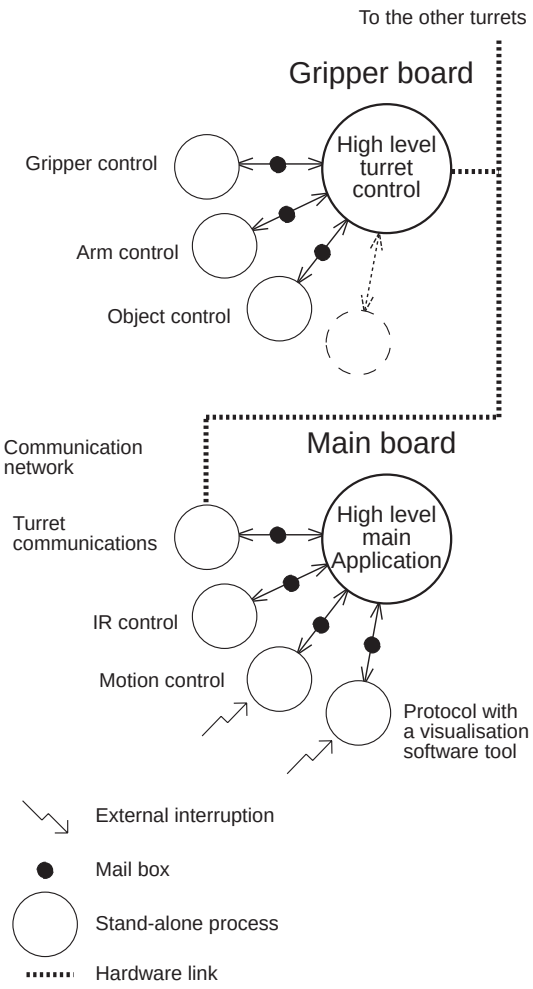

Figure 5. Hierarchical software structure.

3. Software

Managing all the Khepera resources is a complex task. The large number of asynchronous events to control, and the necessity to share some critical interfaces led to the development of a complete low-level software organised as a collection of basic I/O primitives [13].

3.1. Hierarchical software structure

The multi-microcontroller approach and the complex tasks to manage required a hierarchical approach to the software structure. The concept chosen applies when intelligent turrets (equipped with a microcontroller) are used. Two software structures are implemented: a single high-level application program and a number of stand-alone local processes (figure 5). Stand-alone local processes (e.g., for IR sensor sequencing, motion control, wheel incremental-sensor counting, etc.) are executed cyclically according to their own event timer and possibly in association with external interruptions. The high-level application software run the control algorithm and communicate with the stand-alone local processes via a mailbox mechanism. This decoupling of low- and high-level tasks makes the development of complex applications quick and easy.

3.2. Control of Khepera

Experiments with Khepera are performed in two different ways: by running algorithms on autonomous robots or in connection with visualisation software tools.

As already mentioned, the details of the basic input/output activities are managed through a library of stand-alone processes. During the development, the standard RS232 link is used, through a generic high level protocol, to communicate with these processes from a workstation. The application software is therefore run on the workstation and calls to the basic primitives make it possible to monitor the robot activity possible. All standard and specialised visualisation tools can be employed to simplify the control algorithm debugging.

Because the application software is written in standard `C` language, debugged algorithms can easily be converted to run on the Khepera CPU. Applications can be downloaded to Khepera and the robot becomes autonomous from the development environment.

4. Experimentation environment

The quality of the measurements obtained in robot experiments and the efficiency of the whole experimentation process critically depends on the structure of the working environment. Tools currently available for simulation are far better developed than those used for experimenting with real robots. The real time interaction with the control process and the continuous visualisation of the parameters make possible a faster and better understanding of the mechanisms involved. For these reasons, it is necessary to develop better visualisation and interactive software tools adapted to the experimentation tasks.

The simplest way to implement a comprehensive graphical interface is to use a scientific workstation. This must be connected to the robot to collect the data for display and to communicate the orders coming from the user. The physical arrangement of all elements involved in the experiment must be compact, allowing a complete and comfortable control. Thanks to miniaturisation, this can be obtained as illustrated in figure 6: the entire configuration, including robot, en-

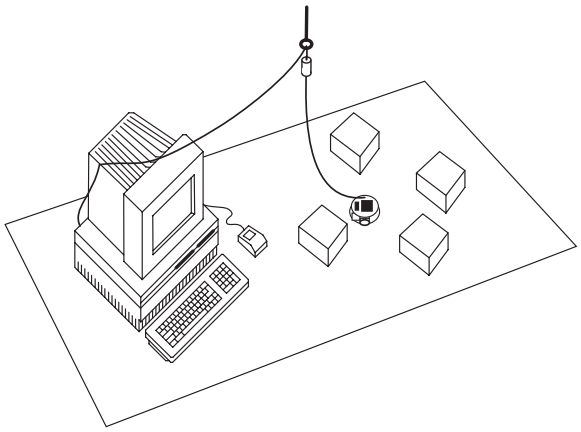

Figure 6. Khepera experimentation environment.

vironment and workstation, is conveniently arranged on a normal table. In the arrangement shown in figure 6 the serial link cable does not disturb the movement of the robot. A device to prevent the cable from rolling up is placed at mid-length on the serial cable. For experiments involving more than one robot, the wired serial link can no longer be used. Special radio communication modules are being developed for this purpose. This additional module will provide means to control several Khepera at the same time.

With a wired or a radio serial link, the data flow between workstation and robot must be as little as possible. To minimise this flow without restricting user ability, the control algorithm runs on the workstation and communicates to the robot only the data concerning sensors and motors. This configuration is optimal when an important number of parameters must be displayed and controlled.

Several programming and visualisation tools are used to perform experiments with Khepera. Here, three programming styles will be presented: the first uses a classical programming language to build standalone applications, the second a complete graphical programming interface and the third is a compromise between the previous two, making the best of both.

4.1. Complete applications

A first programming possibility is to code the control algorithm and the user interface in a traditional way. This is a good choice for software engineers or researchers who have already developed a visualisation and control interface, for instance in a simulation environment. It is often the case that, when a researcher starts to perform real robot experiments, a simulator has already been developed and used for preliminary studies. In this situations, the simulator can easily be adapted by replacing the simulated actions with calls to the interface with the real robot. This can usually be made without modifying the user interface.

Some very interesting results have been achieved with this approach on the neural networks simulator developed by Ph. Gaussier [14]. The simulator is used as a tool to design neural networks for robot control. A real time visualisation interface permits a verifiable step-by-step learning process on the robot.

A similar experience in interfacing Khepera with a simulator is in progress using the simulator BugWorld, developed at the Institute für Informatik of the University of Zürich. In this case, the control interface will be complemented with a measurement system which enables the user to plot the trajectory of the robot in real time on the host screen.

4.2. LabVIEW

The software package *LabVIEW* is a commercial product from National Instruments [15] and runs on several host machines (PC, Macintosh or Sun workstations). LabVIEW has been developed as an environment for the design of *virtual instruments* (VI). Every VI comprises a control panel and an interconnection diagram. On the panel, the user can interactively specify graphical devices for input (e.g., sliding cursors, buttons, text controls) and for output (e.g., gauges, images, graphs). In the diagram, the user graphically enters the functionality of the VI. A library of standard functionalities is available to perform this task: icons performing numerical functions, string treatment, matrix computations, etc. can be linked together to design an algorithm. An icon can be associated with a complete VI and used hierarchically in the diagram of another instrument, thus allowing a modular approach. Moreover, modules can be written in standard programming languages, such as `C` or `Pascal`.

An sample experiment is shown in figure 7. The diagram represents the computation of a subsumption-based algorithm [1]. Two modules, `collide` and `turn wall`, take inputs from the sensors (bottom left icon) and are connected to feed appropriate commands to the motors (right icon). The `sensors` and `motor` icons communicate with the robot through the wired serial link. The two graphs on the panel visualise the state of one motor and of one sensor. The modules in the top right part of the diagram evaluate the period required to recompute the algorithm; this is displayed at the bottom left of the panel.

LabVIEW is an optimal tool for the design of experimentation environments without the use of programming languages. The complete graphical inter-

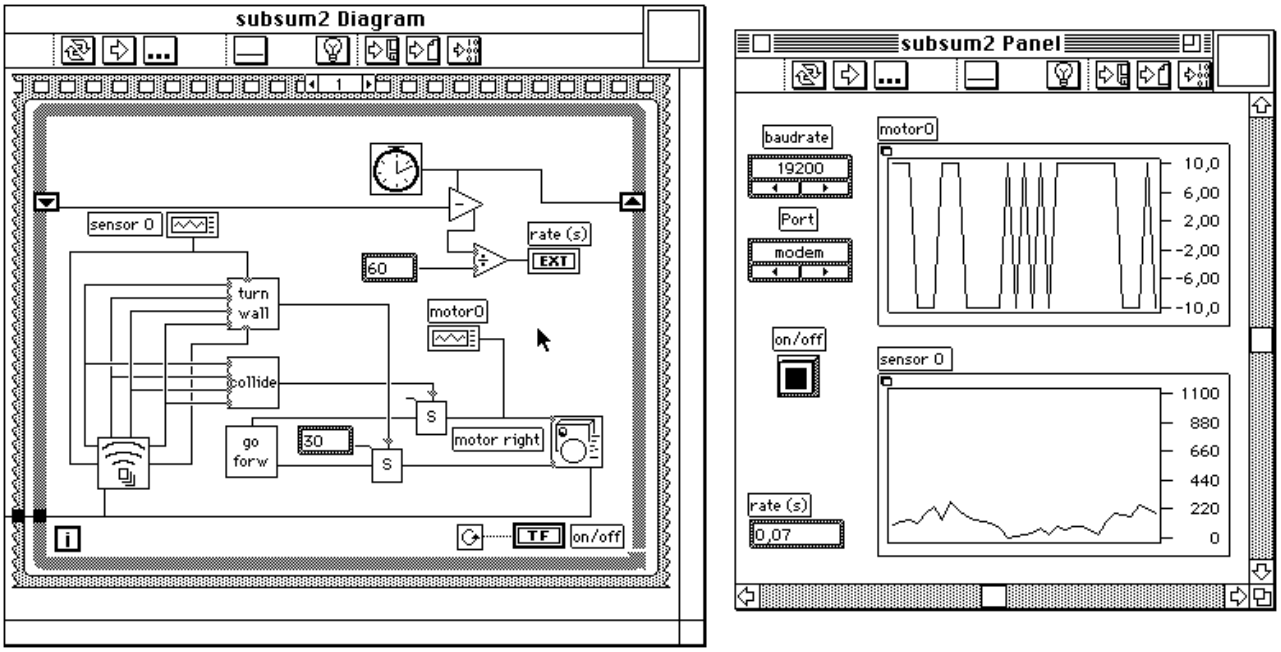

Figure 7. LabVIEW display.

face helps specifying the interaction items of the panel but becomes inefficient when designing complex control algorithms. In this case, it is more efficient to design modules using standard programming languages. The only disadvantage of LabVIEW version 2 is that the display possibilities are somehow limited. The version 3 will provide more interactive capabilities and will be a better design tool for mobile robot experimentation.

4.3. Grapher

Grapher [16] is an experimentation tool developed at LAMI by Y. Cheneval and L. Tettoni for the Esprit Elena Project. Due to code optimisation and to improve performance, the software package is available only on SUN SparcStations. In the Grapher environment, an experiment is defined interconnecting modules that perform sub-tasks such as pure computation, visualisation or control. The programming of these modules is done in `C` language. The interconnections are graphically specified by the user. The wiring diagram is on a single level, therefore preventing a hierarchical approach. For this reason, and to avoid over-complicated wiring schemes, the modules perform quite complex tasks. To facilitate the development, a large number of standard modules are available, making the best of the available hardware possibilities; the performance and flexibility of the visualisation is particularly impressive. Comparing Grapher to LabVIEW, the former is less intuitive, needs some programming knowledge but make complex experimentation efficient. The experiment described in the next section illustrates this aspect.

5. Experimentation in Distributed Adaptive Control

As an example of the development techniques outlined above, the environment to evaluate a control architecture will be presented in this section. The control mechanism is developed according to the design methodology of *distributed adaptive control* [17]. This approach is in turn derived from a distributed self-organising model of the behavioural phenomenon of classical conditioning [18] [19]. The example involves an autonomous agent that can learn to avoid obstacles using collision and proximity sensors.

The control structure consists of a neural net with three groups of neurons (figure 9) named *Unconditioned Stimulus* (US), *Conditioned Stimulus* (CS) and *Motor actions* (M). Neurons of the US group are directly connected with collision sensors, simulated here by the saturation of the proximity sensors. A prewired connection between the US and the M groups implements the robot basic reflex of avoiding obstacles at the time of a collision. Neurons of the CS group obtain their inputs from the proximity sensors. The learning

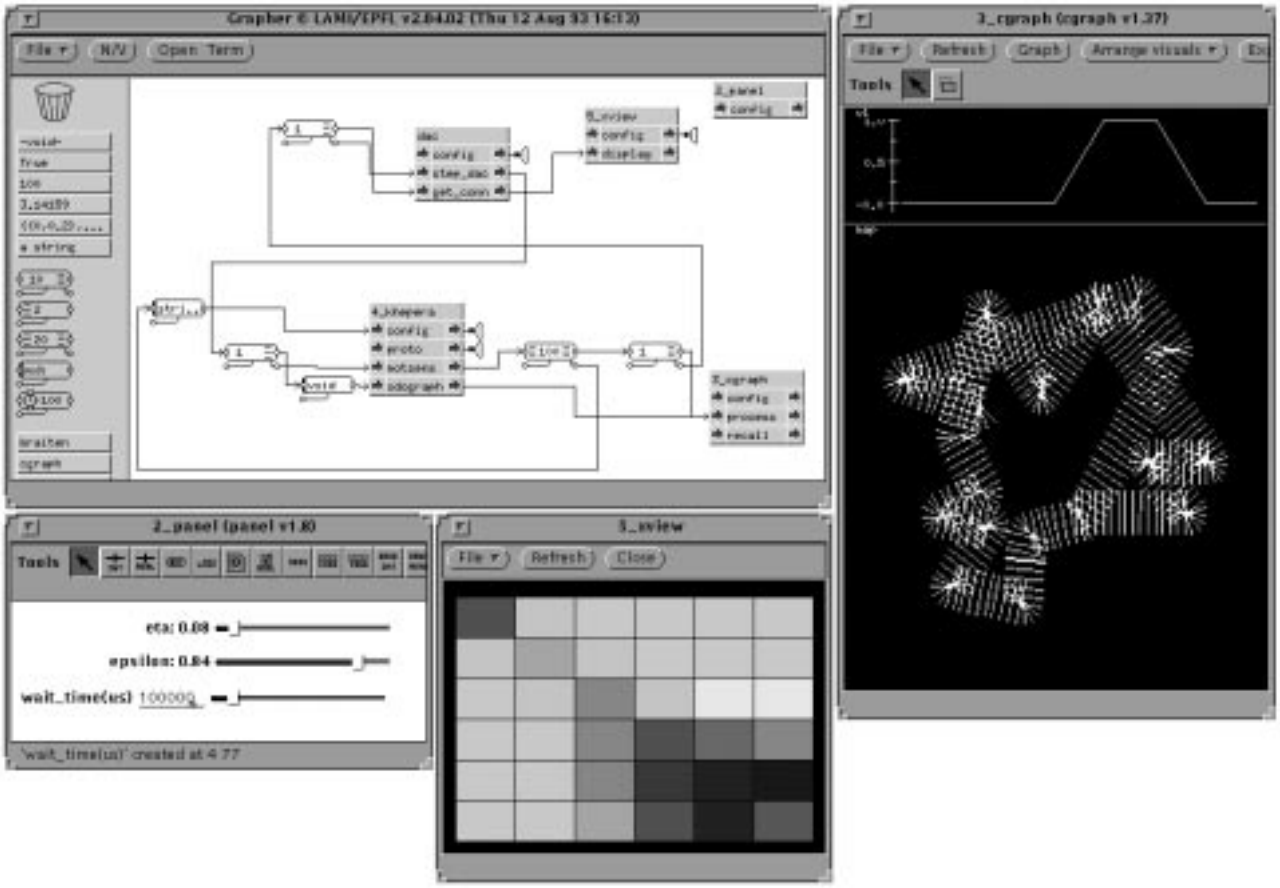

Figure 8. Grapher display.

is performed with an Hebbian rule on the connections between CS and US. The weights $K_{i,j}$ of these connections are updated according to:

$$\Delta K_{i,j} = \frac{1}{N}\left(\eta \cdot s_i s_j - \epsilon \cdot \overline{s} \cdot K_{i,j}\right) \qquad (1)$$

where N defines the number of units in the CS, η is the learning rate, ϵ the decay rate, and $\overline{s}$ the average activation in the group US.

This way, the robot can develop a conditional response, learning to avoid obstacles using the proximity sensors without producing collisions. During the experimentation it is interesting to observe the evolution of the learning process on the matrix K, which depends on η and ϵ.

The software environment used for this experiment is Grapher (see section 4.3). Figure 8 shows the experiment display on a SUN SparcStation. The principal window, on the top left, illustrates the functional diagram of the experiment: The `dac` module (centre top) performs the computation of the algorithm and interacts with the module `khepera` (centre bottom) to control the robot. Three other modules permit user interface. The `panel` module (top right) allows the user to control η, ϵ and the algorithm computation period by means of sliding cursors. The `xview` module displays the K matrix in the centre bottom window. Finally, the `cgraph` module (bottom right) displays the sensor state and the trajectory of the robot, as visible in the rightmost window.

If the control algorithm `C` source with no display capabilities is available, the experimental set-up can be designed in less than one day. The user gains complete control of the physical robot, the development environment and all the parameters of the algorithm in real time, thus obtaining an optimal visualisation of the process.

6. Conclusions

The miniaturisation of Khepera makes a compact and efficient experimentation environment possible. Associated with effective software tools, this robot is an

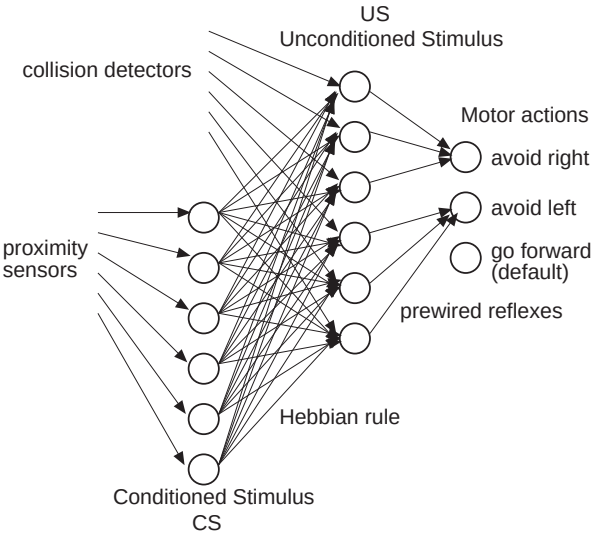

Figure 9. Distributed Adaptive Control experiment architecture.

optimal platform to test control algorithms. The modularity at the hardware, software and control tools levels gives to the user the necessary flexibility to perform accurate experiments quickly. An example of experimentation environment has been presented. The reduced size and cost of the miniature robots described make possible experimentation on collective behaviour among groups of robots. This will be the main research activity in the near future.

Acknowledgements

The authors would like to thank André Guignard for the important work in the design of Khepera and Jelena Godjevac, Paul Verschure, Claude Touzet, Philippe Gaussier, Stéphane Zrehen, Yves Cheneval and Laurent Tettoni for help in testing the algorithms and in the development of the experimentation tools. This work has been supported by the Swiss National Research Foundation (project PNR23).

References

[1] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Robotics and Automation*, RA-2:14–23, March 1986.

[2] J. Heller. Kollisionsvermeidung mit fuzzy-logic. *Elektronik*, 3:89–91, 1992.

[3] U. Nehmzov and T. Smithers. Using motor actions for location recognition. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 96–104, Paris, 1991. MIT Press.

[4] N. Franceschini, J.-M. Pichon, and C. Blanes. Real time visuomotor control: From flies to robots. In *Proceedings of the Fifth International Conference on Advanced Robotics*, pages 91–95, Pisa, June 1991.

[5] R. D. Beer, H. J. Chiel, R. D. Quinn, K. S. Espenschied, and P. Larsson. A distributed neural network architecture for hexapod robot locomotion. *Neural Computation*, 4:356–65, 1992.

[6] J. C. Deneubourg, S. Goss, N. Franks, A. Sendova, A. Franks, C. Detrin, and L. Chatier. The dynamics of collective sorting: Robot-like ant and ant-like robot. In J. A. Mayer and S. W. Wilson, editors, *Simulation of Adaptive Behavior: From Animals to Animats*, pages 356–365. MIT Press, 1991.

[7] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–59, 1991.

[8] F. Mondada and P. F. M. J. Verschure. Modeling system-environment interaction: The complementary roles of simulations and real world artifacts. In *Proceedings of the Second European Conference on Artificial Life*, Brussels, 1993.

[9] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990. Special issue.

[10] L. Steels. Building agents out of autonomous behavior systems. In *The Biology and Technology of Intelligent Autonomous Agents*. NATO Advanced Study Institute, Trento, 1993. Lecture Notes.

[11] Nomadic Technologies, Inc., Palo Alto, Calif. *The NOMAD Robot*. Data-sheet.

[12] P. Dario, R. Valleggi, M. C. Carrozza, M. C. Montesi, and M. Cocco. Microactuators for microrobots: A critical survey. *Journal of Micromechanics and Microengineering*, 2:141–57, 1992.

[13] E. Franzi. Low level BIOS of minirobot Khepera. Internal report R93.28, LAMI - EPFL, Lausanne, 1993.

[14] P. Gaussier. *Simulation d'un système visuel comprenant plusieurs aires corticales: Application à l'analyse de scènes*. PhD thesis, Paris XI - Orsay, Paris, November 1992.

[15] National Instruments Corporation. *LabVIEW 2*, January 1990. User Manual.

[16] Y. Cheneval, P. Bovey, and P. Demartines. Task B2: Unified Graphic Environment. Delivrable R1-B2-P, ESPRIT Elena Basic Research Project no. 6891, June 1993.

[17] P. F. M. J. Verschure, B. J. A. Koese, and R. Pfeifer. Distributed adaptive control: The self-organization of structured behavior. *Robotics and Autonomous Agents*, 9:181–96, 1992.

[18] P. F. M. J. Verschure and A. C. C. Coolen. Adaptive fields: Distributed representations of classically conditioned associations. *Network*, 2:189–206, 1991.

[19] I. P. Pavlov. *Conditioned Reflexes*. Oxford University Press, London, 1927.