Modeling system-environment interaction: The complementary roles of simulations and real world artifacts

Francesco Mondada[1], Paul F.M.J. Verschure[2]

[1] Laboratoire de microinformatique
Ecole Polytechnique Fédérale de Lausanne
INF-Ecublens, CH 1015 Lausanne
Switzerland.
e-mail: mondada@lamisun.epfl.ch

[2] AI lab, Institute for Informatics, University of Zürich,
Winterthurerstrasse 190, CH- 8057 Zürich,
Switzerland.
e-mail: verschur@ifi.unizh.ch

Abstract

In the study of adaptive behavior an important issue has become the possible dichotomy between simulations and real world artifacts (robots). In our work we try to find a synthesis between these two methods to study the interaction between autonomous agents and the real world. The strategy we propose is one that initially relies on simulation studies to develop and test the basic properties of the control architecture. In a second step this control architecture is validated against the real world using robots. When this phase is successfully completed other experiments with the robot can be performed that generalize to new domains. As an illustration of the first phase we present the implementation of Distributed Adaptive Control (DAC) on a real robot. The miniature experimental mobile robot Khepera and the development environment are presented. As an illustration of the second phase the robot is equipped with a simple visual system incorporating some obvious properties of the retina, topology and band pass filtering. The behavior and properties of the control architecture in this set up will be evaluated.

1. Introduction

A relatively new trend in artificial intelligence is to place the study of intelligence in the perspective of autonomous agents interacting with the real world (e.g., Brooks, 1991). In earlier work we have argued that merely building autonomous agents, however, does not automatically solve the fundamental problems of AI (e.g., Pfeifer and Verschure, 1992) and that the central issue is one of the ontologies involved (Verschure, 1992); the issue of "doing the right thing" should be addressed in the perspective of the agent and not of the designer/observer. In traditional AI the issue might have been how one should predefine a system in such a way as to perform well on a certain task. In our case we focus on the question how a self-organising process can lead to adaptive behavior (See also Edelman, 1989).

The complexity of the dynamics of system-environment interaction can be seen as an obstacle to the development of this approach. To model the interaction between an autonomous agent and its environment one either relies on simulations or on hardware implementations. Often these two methods are taken to be in conflict. On one hand simulations do not allow the detailed study of the physics of the world and of the properties of the system (sensors and effectors). In many cases one can doubt their plausibility. For instance, if one relies on the simulation of an agent moving about in a discretized world. On the other hand systematic explorations of control

architectures on real robots require a large time investment and systematicity is not guaranteed. This method often lacks the flexibility one needs for such an exploration. To overcome these problems we strive towards the integration of both methods.

In this paper we want to give an example of how simulation studies and experiments with real robots are both necessary in the process of developing and understanding control architectures. We distinguish two phases in the development of a control architecture. In the first one the control architecture is evaluated using simulation studies and validated against the properties of the real world using a robot. In this phase the functionalities of both systems are similar. The simulation study serves the purpose of understanding the dynamics of the control architecture. It can not provide a final test on its real world plausibility. This can only be achieved by using real world artifacts. In the second phase experiments with the real robot can be made in new domains independent of simulation studies in order to explore how the control architecture generalizes to other characteristics of the real world and other sense-act systems.

We will illustrate this methodology using a control architecture developed according to the design methodology of distributed adaptive control (DAC) (Verschure et al., 1992). The robot experiments will be performed using the platform developed by the first author, the miniature mobile robot Khepera. We will show that the results of the simulations and the robot in the first phase converge. These results are not only an indication of the robustness of the control architecture, but illustrate the importance of an integrated methodology for the design and testing of autonomous agents and their control architecture. Simulations are performed for the exploration of the control architecture while hardware experiments provide a final validation against the real world. The power of simulation studies in understanding control architectures was, for instance, demonstrated in Almassy and Verschure (1992) where the behavior of DAC over a large parameter space was explored using genetic algorithms. In this case about 1000 individual configurations were examined to show the robustness of the control architecture for a large set of parameter settings. Hardware experiments, however, allow an evaluation of a control architecture that include properties of the sense-act system and of the real world that are not that easy to simulate. As an example of these virtues of hardware experiments we will discuss the results of experiments with Khepera using a simple vision system.

2. The hardware platform: the mobile robot Khepera

2.1 Design concept

Khepera is a miniature mobile robot, especially designed for testing control algorithms. In its design the flexibility of the platform was one of the central concerns. The goal was to develop a system that on one hand does not require a lot of space to perform experiments and on the other has sufficient computational power with a low general energy consumption.

The elementary configuration has a cylindrical shape with a diameter of 55 mm and a height of 30 mm (Figure 1). It consists of two modules: the base plate and the processor module. Since all

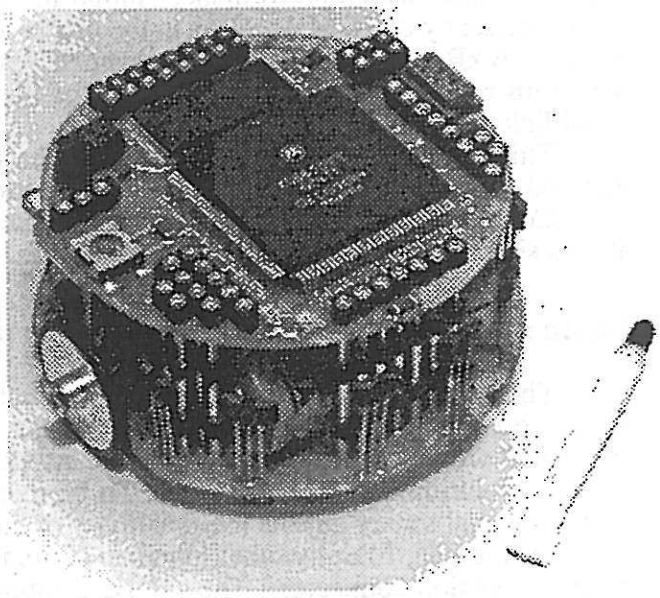

Figure 1: The miniature robot Khepera

modules are interconnected by an extension bus the system can be expanded by other modules, like the "vision" module described in section 2.4, added in piggy-back. The small size does not allow the robot to carry a lot of equipment around, but it permits the user to perform experiments on a relatively small space (in our case a field of .25 m^2).

2.2 Base plate

The base plate is situated at the bottom of the robot. It constitutes the elementary interface with the real world: robot motion and obstacle or light detection.

The robot uses two wheels for locomotion. Every wheel is driven by a DC motor. An incremental encoder is used to detect the revolutions of the motor. In the middle of this module four chargeable batteries are located. The robot can operate autonomously on these batteries for about 20 minutes.

Obstacles and light are detected by 8 infra red send-receive sensors. These sensors allow the detection of emitted infra red light or of ambient light. At the front of this module six sensors are placed and two are placed at the back. The combined receptive fields of the six front IR sensors spans over -90° and 90° from the centre of the robot. The angular resolution of each sensor is relatively big, about 50°.

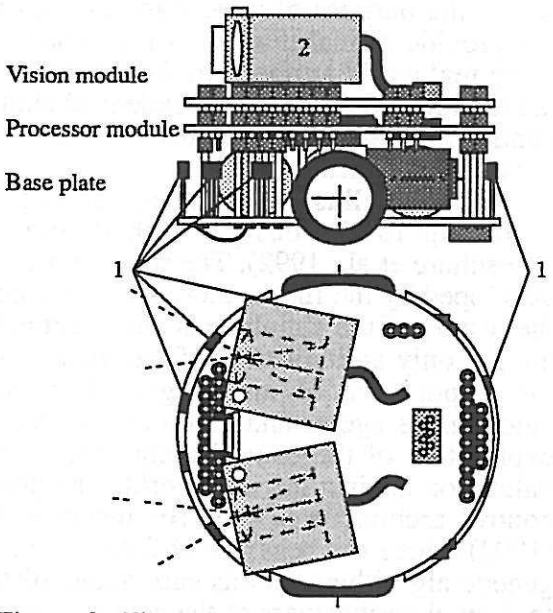

Figure 2: The robot extended with the "vision" module. 1: location of IR sensors. 2: camera.

2.3 The processor module

The processor around which this module is developed is a Motorola 68331 microcontroller with a performance similar to the Motorola 68020 processor. It has a clock frequency of 16 MHz. Next to the processor this module contains 256 kByte of RAM, 256 kByte of ROM, six A/D channels with a resolution of 10 bits, and a serial port with a maximal transmission speed of 38400 Baud. The cable supporting the serial link can also be used for power supply.

This module is plugged on top of the base plate using an array of connectors that make the electrical and mechanical connections.

Connecting this module with the base plate constitutes the basis of the robot and allows simple navigation and obstacle detection.

2.4 An extension: the one-dimensional vision module

The one-dimensional vision module has been designed to allow experiments with more sophisticated sensors. It consists of two one-dimensional cameras (indicated with 2 in Figure 2). Every camera delivers a linear image of 64x1 pixels with a resolution of 10 bit per pixel. To be independent of the intensity of ambient light, a special sensor adjusts the sensitivity of the cameras.

This module, like any other imaginable module, is added to the basis of the robot by the extension connectors. The extended robot and the positions of the sensors are displayed in Figure 2.

2.5 The development environment

The development environment is running on a workstation that communicates with the robot through a wired serial link. This allows the development of the software on the workstation in a standard C environment. The compilated code is down loaded on the robot through the serial link and executed. After that the serial link is available for communication with the control process running on the robot. This communication is facilitated by the size of the robot: the table on which the workstation is placed is usually large enough to perform experiments with the robot.

3. Distributed Adaptive Control

The control architecture we evaluate is developed according to the design methodology of distributed adaptive control (Verschure et al., 1992, in this paper also a comparison with other approaches is made) which is derived from a distributed self-organising model of the behavioral phenomenon of classical conditioning (Verschure and Coolen, 1991). We will base our example on an agent that can learn to avoid obstacles and approach targets (light sources).

The basic setup of the agent is given by its value scheme (Edelman, 1989) which can be seen as genetically predefined. The value scheme defines the properties of the sensors and effectors, the morphology of the system, the structure of the control architecture, and the mechanisms for changing the properties of this structure. Moreover, it defines some basic reflexes, like if the system collides to the right it will reverse and turn to the left. These reflexes consist of prewired relationships between primitive sensors of the system and its actions. They provide the system with a coarse adaptation to the environment. The way in which they are activated by the system-environment interaction will initially completely determine the actions the agent will execute. To enable the system to adapt to the exact properties of its interaction with the environment it is equipped with a distal sensor which responds to properties of the environment extended beyond the morphology of the system. The integration of this sensor in the actions of the system will lead to a fine tuned adaptation to the system-environment interaction. This integration process can be seen as the development of a specific categorisation of this interaction (see Verschure and Pfeifer, in press, for a further analysis of these categories). It is important to note that the control architecture defined by the value scheme must be seen as a structure with specific spatial properties.

Sensors: The agent has two types of sensors: The distal sensors and the proximity ones. In the first experiment this first category of sensors is based on the IR signals and in the second experiment on the signal stemming from the two cameras. The proximate sensors, collision sensors and target sensors, are respectively defined by the saturation of the IR sensors and the detection of ambient light by the IR sensors.

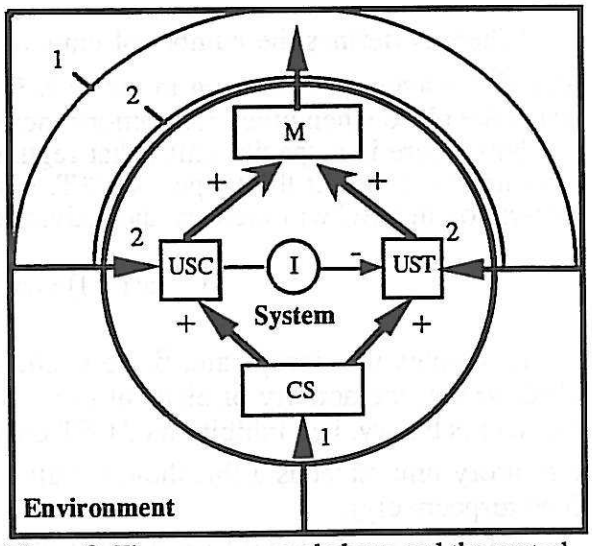

Figure 3: The sensors, morphology, and the control architecture of the agent: 1, receptive field of the distal sensor. 2, area covered by collision sensors and target sensors. CS receives input from the distal sensor. USC receives input from the collision sensors. UST receives input from the target sensors. M is the group of command units. I is the inhibitory unit. + indicates an excitatory connection, - represents an inhibitory connection.

Actions: The agent can perform 4 actions: "turn left", "turn right", "reverse", and "advance". The avoidance actions consist of a "reverse" action followed by a turn. The approach actions consist of only a turn. The default action of the system is to advance. The motor activity in turn actions is fixed, in the hardware experiments current is applied for 5 milliseconds.

Control: The control architecture consists of four groups of units and one inhibitory unit. The 6 units of the first group (CS - Conditioned Stimulus) receive their input from one of the IR sensors. The continuous activation, s_i, of each unit i is determined by the normalised value of the intensity of the infra red light reflected by the obstacles and detected by IR sensor r_i. In this way these units code a measure for 'time to contact' (Lee, 1976).

The units of the second group (USC - Unconditioned Stimulus Collisions), which also consists of 6 units, receives its input from the collision sensors. As in the case of CS there is a one to one mapping between units of USC and collision sensors. If one collision sensor is triggered, which happens when an IR sensor saturates, the corresponding unit of USC will receive an input of 1. The third group (UST - Unconditioned Stimulus Targets), also consisting of six units, relates to the intensities of the ambient light detected by the IR sensors. Next to the input from a specific primitive sensor the units of both these groups also receive input from CS. This input is modulated by the weights of the projections from CS to USC and UST. The input or local field, h_i^λ, to the units of USC and UST is defined as:

$$(1) \quad h_i^\lambda = c_i^\lambda + \sum_{j=1}^{N} K_{ij}^\lambda s_j$$

Here λ indexes the groups USC and UST, c_i^λ denotes the input from the related proximity sensor, $c_i^\lambda \{0, 1\}$, s_j the activation of the units in CS, s_j [0, 1], and K_{ij}^λ the weight of the projections of CS to UST and USC. These weights are updated according to:

$$(2) \quad \Delta K_{ij}^\lambda = \frac{1}{N}\left[\eta \; s_i^\lambda s_j - \varepsilon \bar{s}^\lambda K_{ij}^\lambda\right]$$

Where N defines the number of units in CS, η the learning rate, ε the decay rate, and $\bar{s}^\lambda$ the average activation in group λ. $\bar{s}^\lambda$ introduces an active decay: decay will only take place when other connections increase in strength.

Next there is a special unit I that regulates the relation between USC and UST. This unit will inhibit the output of UST. It will not inhibit the activity of this group. Activation in USC will increase the activation, a, of I.

$$(3) \quad a(t+1) = \alpha a(t) + \beta\left(\frac{1}{N^c}\sum_{i=1}^{N^c} S_i(t)\right)$$

α denotes the decay rate, β the excitation rate, N^c the number of elements of USC, and s_i the activity of element i of USC. The output function of the inhibitory element is binary, i.e., inhibition of UST only takes place if the activation value of the inhibitory unit exceeds a threshold (in all experiments α and β were set to 0.9 and 0.99 respectively).

The actions are coded by a set of command units in group M which consists of 5 elements.

The relations between the UST, USC and M are prewired. A collision to the left will automatically trigger a "reverse-and-turn-right" action (and symmetrically for collisions to the right). If a target is detected to the left of the system a "turn-left" action will be executed. If a target is detected to the right the system will turn to the right. One unit in USC or UST can trigger a unit in M. The groups USC and UST can be defined in two clusters of units each dependent on the unit in M to which they

project: e.g. all units of USC that are connected to collision sensors located to the left of the center of the agent project to the "reverse-turn-right" command unit.

Since the activation of all units in the system can only have positive values learning is expressed in the development of excitatory connections between CS and USC and UST. Also the connections between USC and UST and M are excitatory. The inhibitory unit I, however, implements a specialized inhibitory circuit.

4. Phase I: The transfer from simulations to hardware

The results we report on the transfer from the simulation study to the real world artifact focus on avoidance behavior. The full architecture reported in the last section was, however, implemented on the robot.

The configuration of the obstacles used in this first experiment is showed in Figure 4. This environment (of 50 x 50 cm) has been build with little pieces of wood on the table near to the computer collecting and showing the results of the experiment. The robot was connected with the computer with a cable for communication and power supply.

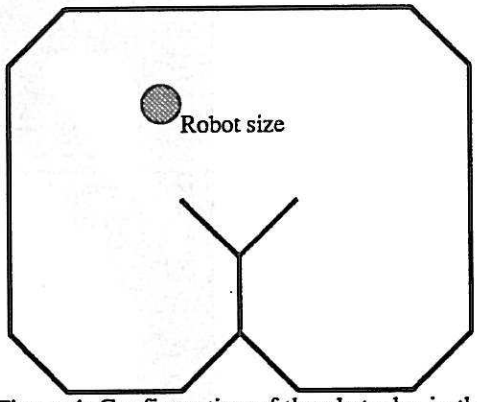

Figure 4: Configuration of the obstacles in the first experiment

The program that was downloaded on the robot was the same C source code as used in the simulations with some modification for the interfacing with the sensors and the motors. Also the same parameters were used as in the simulation (all thresholds were set to 0.5, $\eta = 0.1$, $\varepsilon = 0.9$, see Verschure & Pfeifer, in press, for a complete description). The world simulation is replaced by some low-level routines that collect the sensor data (distance measurement) and drive the motors (velocity regulation). The time segmentation is the same as in the simulation: A sequence of steps is performed. At every step the robot reads the sensor values, updates the activations of the network, executes an action, and adapts the weights of the network. Only the default action, going forward, is continuously performed until it is interrupted by an event on the sensors.

As a quantification of the behavior Figure 5 depicts the accumulated amount of collisions over time for three different experiments which each lasted about 10 minutes.

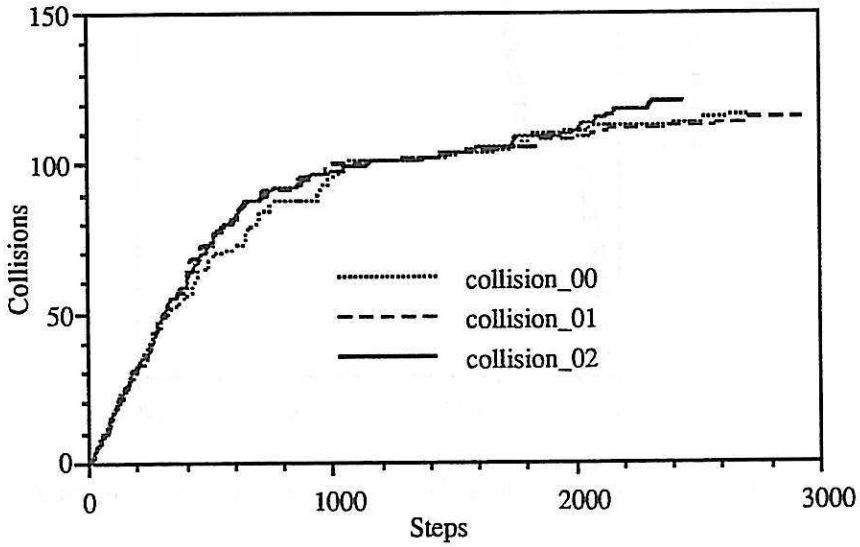

Figure 5: Total amount of collisions of the robot after a given number of steps

These learning curves show an equivalent development as reported in earlier work on this model (e.g., Verschure et al., 1992). To show the parallel between the properties of the robot and the simulated agent the same experiment was performed using the simulator. In this case CS is organised following a one-to-one mapping with 37 elements of a range finder spanning the same receptive field. Onto USC the states of 37 collision sensors are projected (See Verschure et al., 1992, for a complete description).

Figure 6 shows the trajectory followed by the agent in 1000 steps.

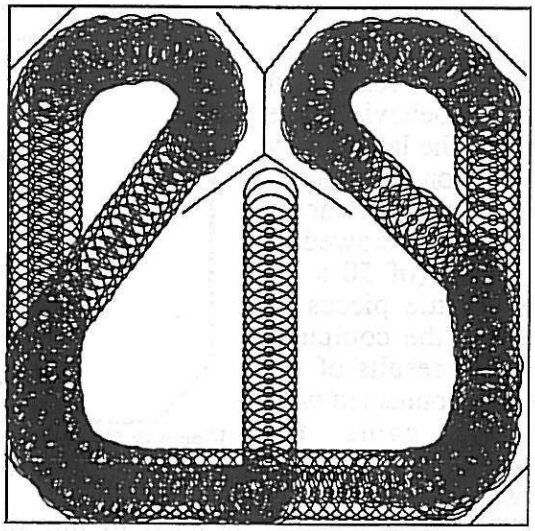

Figure 6: Example of a simulated trajectory

Qualitatively the robot and the simulated agent showed similar behavior. The robot, however, displayed a higher variability in the specifics of the actions that make up the trajectory; e.g., while the simulated agent will always exactly turn over the specified turn angle (9°) the robot is not and cannot be that precise.

Figure 7 shows the learning curve of the simulated agent.

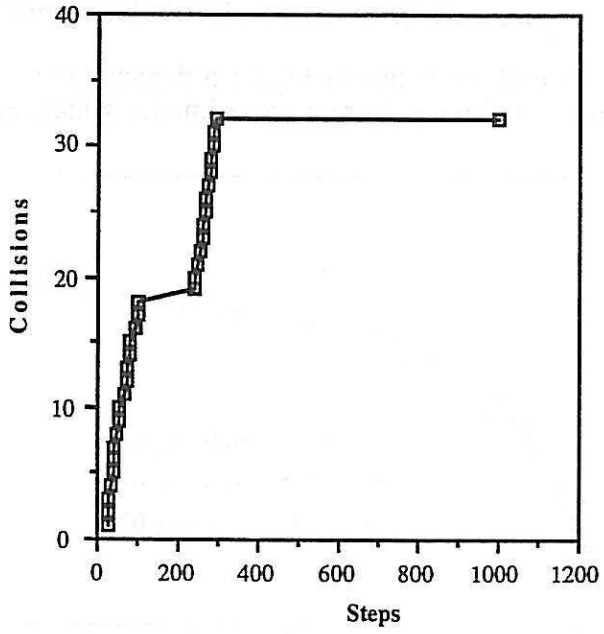

Figure 7: Total amount of collisions of the simulated agent after a given number of steps

When we compare the learning curves of the robot (Figure 5) with those of the simulated agent the overall shape shows a strong similarity. The robot, contrary to the simulated agent, never reaches a state where no collisions are encountered. This can be explained out of the earlier mentioned higher variability in its behavior. Where the simulated agent displays 'ideal' behavior and also settles into a rather stable trajectory, the robot is confronted with the noisiness of the interaction with the real world. These results indicate that the transfer from simulations to hardware experiments in this task environment was successfully performed.

5. Phase 2: Experiments with a simple visual system

5.1 Layout of the visual system

One of the characteristics of the DAC control architecture is its independence of the exact properties of the sensors. To explore this property we have performed experiments where the distal, IR, sensors were replaced by another device: a one-dimensional visual system. The hardware module added to the robot to allow this experiment has been described in the section 2.4.

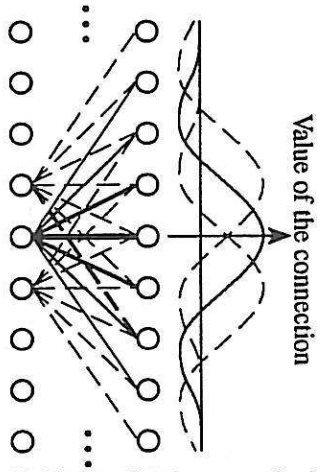

Figure 8: Connection between the input layer and the filtered layer.

The visual system is designed to detect and differentiate several different spatial frequencies present on the horizontal vision line of the two eyes. The system consists of several layers of units. We consider the array of pixel values as the input layer of the system. Every input is connected to the units of a frequency specific layer by connections having a particular "Mexican hat" form that expresses the properties of a specific on-center-off-surround structure (Figure 8). These connections perform a band pass filter around a base frequency. The value of this base frequency is given by the width of the "Mexican hat" function.

Four of these frequency specific layers are connected in parallel to the input layer. Every layer is excited by a different range of frequencies. The average of the absolute activity of the layer is proportional to the presence of these specific spatial frequencies on the one-dimensional image. One output unit per layer collects this average value. The four output units characterise the spectrum of the image, representing the presence in the image of four categories of frequencies, from high (hf) to low frequency (lf). An example of the activity of the four output units is given in Figure 9.

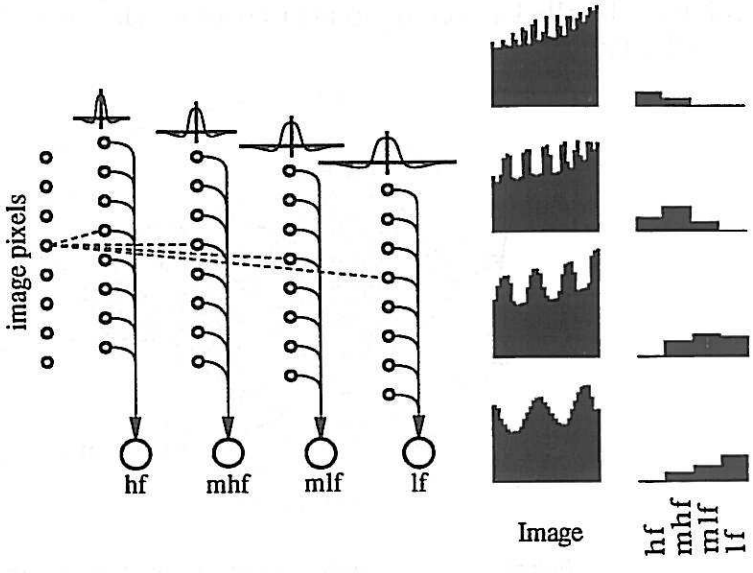

Figure 9: Global structure of the vision system with an example of the activation of the output units.

Within one visual subsystem, camera and neural net processing, the spatial information of the input is lost: its organization on the array of light sensitive cells.

The visual system is only sensitive to certain ranges of frequencies. For instance, shifts of the stimulus on this array cannot be detected. By combining two cameras, however, the input to the system is also categorized in terms of its spatial organization: whether it is the left or the right camera. The network architecture is repeated for every camera. In this way the output of the visual system is coded by eight units.

5.2 Results

The eight output units of the visual system are used as CS in the control architecture. The environment (Figure 10) consists of white walls with regular black vertical lines. This gives the walls a particular horizontal spatial frequency.

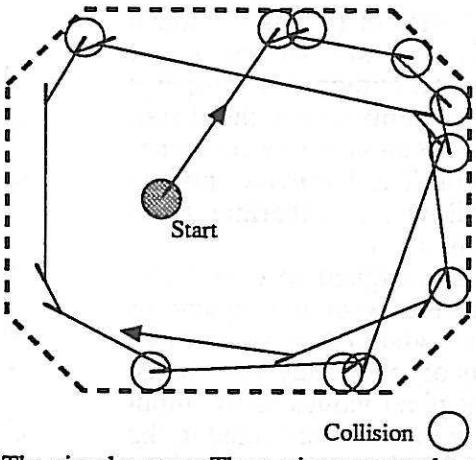

Figure 10: The visual system: The environment and a typical path.

The general results are similar to those obtained in the previous tests made with the IR sensors and in the simulations: the system successfully integrates its visual system into its actions (Figure 10 depicts the trajectory followed by the robot). To further understand the learning process we will focus on the characteristics of the weight matrix, $\mathbf{K}^{\lambda}$, that implements this integration process.

Contrary to Verschure and Pfeifer (in press), where the learning performance was analysed by translating the weights back into world coordinates, we will in this case focus on the values of the weights. The interconnectivity between CS and USC is represented in Figure 11. The collision sensors placed at the left-frontal part of the robot are labelled according to their position relative to the direction of displacement: 5°, 45°, and 85°.

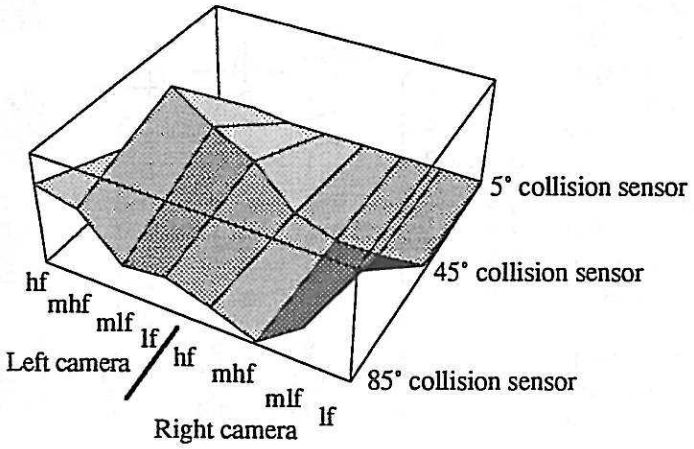

Figure 11: Interconnection matrix between the CS field (vision system output neurones) and the left half part of the USC field.

The middle high frequency (mhf) of the left camera is associated most strongly with USC. In this case with the unit connecting to the collision sensor located at 45°. Given the properties of the robot and the environment this can be explained. Figure 10 already showed that the robot mainly encounters collisions at 45° or more. The angle under which the left camera now detects the stimuli is modulated towards the higher frequencies. The angle under which the right camera detects the stimuli related to the general type of collisions implies a modulation towards the lower frequencies.

6. Discussion

The aim of this paper was to demonstrate the importance of a combined approach in modeling system-environment interaction. Simulations as a means to explore the behavior of control architectures for autonomous agents. Hardware experiments to provide a validation of the model against the real world. The results show a strong convergence between these two methods in the case of distributed adaptive control. Moreover, the independence of the control structure from the exact properties of the sensors was demonstrated. In a next step we will focus on multi-sensor fusion.

Acknowledgement

We would like to thank Edo Franzi and André Guignard for the important work in the design of Khepera. This work has been supported by the Swiss National Research Foundation (project PNR23).

References

Almassy, N., & Verschure, P.F.M.J. (1992) Optimizing self-organizing control architectures with genetic algorithms: The interaction between natural selection and ontogenesis. R. Männer and B. Manderick (Eds.)*Proceedings of the Second Conference on Parallel Problem Solving from Nature*. 451-460. Amsterdam, Holland: Elsevier.

Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence*, **47**, 139-159.

Edelman, G.M. (1989). *The Remembered Present: A biological theory of consciousness*. New York: Basic Books.

Pfeifer, R, & Verschure, P.F.M.J. (1992). Beyond rationalism: Symbols, patterns and behavior. *Connection Science*, **4**, 313-325.

Verschure, P.F.M.J. (1992). Taking connectionism seriously: The vague promise of subsymbolism and an alternative. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 653-658. Hillsdale, N.J.: Erlbaum.

Verschure, P.F.M.J., & Coolen, A.C.C. (1991). Adaptive Fields: Distributed representations of classically conditioned associations. *Network*, **2**, 189-206.

Verschure, P.F.M.J., Kröse, B.J.A., & Pfeifer, R.(1992). Distributed adaptive control: The self-organization of structured behavior. *Robotics and Autonomous Agents*, **9**, 181-196.

Verschure, P.F.M.J., & Pfeifer, R. (in press). Categorization, Representations, and the dynamics of system-environment interaction: a case study in autonomous systems. In: J.A. Meyer, H. Roitblat, & S. Wilson (Eds.) *From Animals to Animats: Proceedings of the Second International Conference on Simulation of Adaptive behavior 1992*.