

In Jin-Kao Hao, Evelyne Lutton, Edmund Ronald, Marc Schoenauer, and Dominique Snyers, editors, selected papers of the 3rd European Conference on Artificial Evolution (AE97), volume 1363 of Lecture Notes in Computer Science, pages 303-313, Nîmes, France, 22. - 24. October 1997. Springer-Verlag 1998.
This work was awarded with the first prize at the international Khepera contest.

The Dynamical Nightwatch's Problem Solved by the Autonomous Micro-Robot Khepera

A. Löffler*, J. Klahold, and U. Rückert

Paderborn University, Heinz Nixdorf Institute, System and Circuit Technology,
Fürstenallee 11, D-33102 Paderborn, Germany
E-mail: loeffler,klahold,rueckert@hni.uni-paderborn.de
<http://hni.uni-paderborn.de/fachgruppen/rueckert>

Abstract. In this paper, we present the implementation, both in a simulator and in a real-robot version, of an efficient solution to the so-called dynamical nightwatch's problem on the micro-robot Khepera. The problem consists mainly in exploring a previously unknown environment while detecting, registering and recognizing light sources which may dynamically be turned on and off. At the end of each round a report is requested from the robot. Therein we made use of an agent-based approach and applied a self-organizing feature map in order to refine some of the behaviour generating control-modules.

1 Introduction

1.1 Motivation

Our basic motivation to consider the implementation of behaviour generating control structures in mobile autonomous systems is represented by the following question: "What is the most complex task a robot is able to carry out, thereby coping with its limited resources, e.g. the available sensor sources, finite energy supply and limited processor power? The encountered consequences may be a restricted perception of its environment, runtime constraints and the problem of severe real-time demands." To be able to successfully envisage this kind of problem, we strongly believe in adopting the following two-step program: firstly, to implement software solutions making recourse to concepts of neural information processing; secondly, to replace single software modules by resource-efficient microelectronic components. In this framework, the contents of the present paper corresponds to the first point, i.e. the implementation of a software solution to a hard robotics' problem. Hereby, the micro-robot Khepera (see Fig.1) serves as an exemplary model for a mobile autonomous system.

Processing the sensor data of a mobile robot appropriately seems still to be a difficult and largely unsolved problem [1]. Hence, it remains an in general challenging task to consistently implement a set of behaviour generating control

* Supported by DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", GRK 124/2-96.

modules based on the different sensor sources of the system in question. Considering the micro-robot Khepera in its basic configuration, there are three main sources of sensor data which may be used to implement such modules:

- | <i>source of sensor data</i> | <i>task of the respective control-module</i> |
|-------------------------------------|-----------------------------------------------|
| (1) the infra-red proximity sensors | → exploration of the environment |
| (2) the ambient light sensors | → detection and registration of light sources |
| (3) the wheel-based step counters | → basic positioning system. |

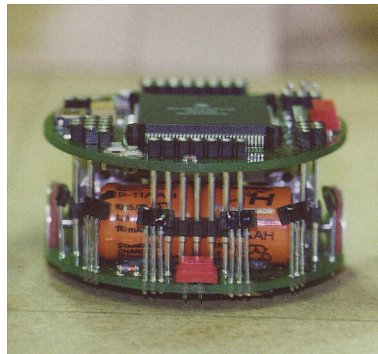


Fig. 1. Khepera is a micro-robot which has a diameter of 55mm, a height of 30mm and weighs 70g. The processor system contains a Motorola 68331 with 256Kbyte RAM and 256Kbyte ROM. Khepera perceives its environment using 8 infrared distance- and light-sensors; moreover incremental encoders placed on each motor axis are available for step counting.

In the next paragraph, we define an appropriate problem, i.e. one which includes all three of the above mentioned tasks.

1.2 The Dynamical Nightwatch's Problem

Imagine the town's nightwatch strolling over central square at late evening, looking for light sources, e.g. street lamps, lighted windows, etc., thereby avoiding obstacles, following walls (because he is a bit short-sighted) and turning round at dead ends. Having found the first light source, he fixes his report-board at the wall next to it to write down everything he will have encountered when he will arrive again at this location. In fact, he notes whether he has found a new lantern or recognized an old one and its current state (on/off) since the last stop at the board. Probably he would try to guess his position on the square by counting his steps and using already registered light sources as local landmarks to verify his counting.

The micro-robot Khepera should be able to perform an analogous task when transferred to a kind of suitable toy-world.

1.3 Design Principles

Animals may be considered as biological analogies to mobile autonomous robots. Other than the latter, they constantly had (and still have) to prove their fitness by surviving a natural selection process. Thus, it seems profitable to exploit their evolutionary optimized strategies for robot design, in this context especially those which deal with information processing. In order to do this, we adopted the following design principles inspired by concepts of neural information processing thereby making recourse to the so called agent paradigm [2-4] and some earlier works in cybernetics [5] :

- simple modules cope with simple tasks (e.g. obstacle avoidance, edge following, etc.)
- the control-modules are organized in parallel, i.e. no explicit hierarchical structure is implemented, and their results are simply superpositioned. In case of this not being possible, a data/event-driven decision is made (e.g. in dead ends, the turning-round-module takes control).
- no global goals are explicitly given to the robot
- arising complex behaviour is thus a result of the interaction of the basic modules
- adaptive algorithms, e.g. neural networks, are used to incorporate acquired informations about the environment.

2 Exploration

Enabling Khepera to explore its environment, we had to ensure free motion, i.e. avoiding obstacles, leaving substructures of the environment and not getting stuck in dead ends or similar structures. This was attained by implementing the following three basic control-modules:

2.1 Obstacle Avoidance

The obstacle avoidance was implemented as in Braitenberg's vehicle IIIb [5] using the very simple neural concept of cross-inhibition. This means that if one of Khepera's infra-red sensors detects an obstacle, the opposite wheel will be slowed down proportionally to the sensor input. By this way the obstacle is avoided (see Fig.2).

2.2 Edge Following

Khepera tries to keep the distance to the wall, i.e. the value of the outmost sensor, constant (see Fig.3). Note that no explicit distance to the encountered wall

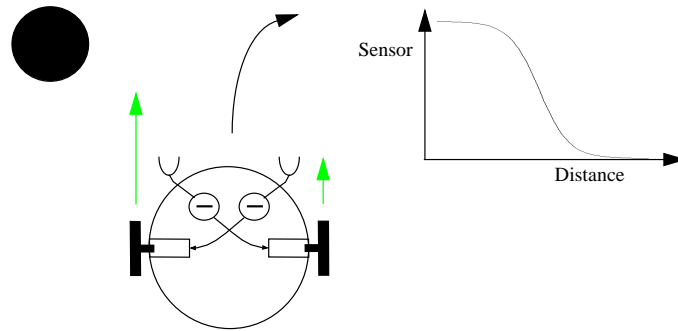


Fig. 2. A sketch of the robot avoiding a cylindrical obstacle (left) and an infra-red sensor vs. distance-to-light source characteristic are shown (right).

is be kept which allows a more flexible behaviour of the robot than otherwise be possible, especially in narrow corridors and similar structures. When a wall is detected for the first time (and avoided by means of the obstacle avoidance module), the outmost sensorvalue is registered. Afterwards, the robot will turn away from the wall, if this sensorvalue increases, and respectively will turn towards it, if it decreases. This algorithm is particularly simple, easy to implement and robust. Edge following allows to leave substructures of the environment quicker than it would be possible otherwise.

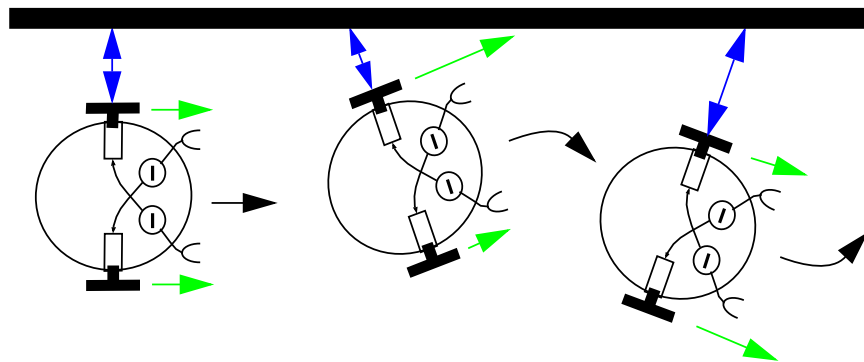


Fig. 3. Visualisation of the simple edge-following algorithm where three different situations might occur: a wall is detected (register sensorvalue_memory), turn away from wall (if sensorvalue > sensorvalue_memory), turn towards wall (if sensorvalue < sensorvalue_memory).

2.3 Turning

If a situation occurs in which Khepera is not able to leave a small area (symbolized by the circle) after some tries, it simply turns round to where it came from (see Fig.4). This module ensures that the robot does not get stuck in dead end like structures.

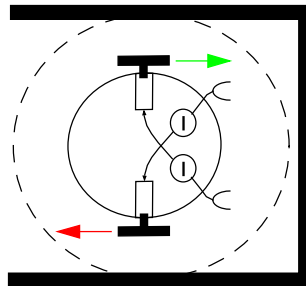


Fig. 4. The turning procedure.

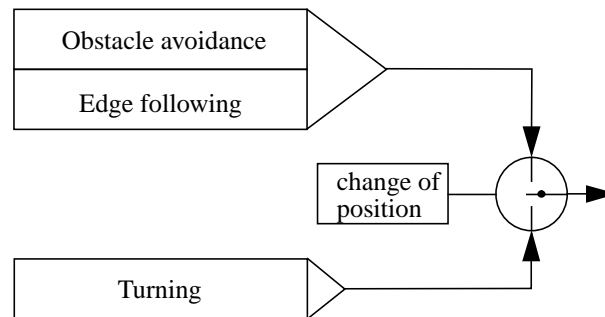






Fig. 5. The interplay of the three control modules ensuring constant exploration.

The interplay of the three modules is shown in Fig.5. A module, recognising the change of the position of Khepera, decides whether obstacle avoidance and edge following or the turning procedure get active. This means that if the robot is not able to change its position sufficiently during some 50 steps, it will turn around thereby suppressing the two other exploration modules; otherwise a superposition of the normal obstacle avoidance/edge following algorithms is used.

3 Positioning

An at least locally reliable positioning system had to be necessarily at basis of the light-source mapping process. For this purpose, we used the step counting functionality of Khepera. The position vector of Khepera consists of an x -value, y -value and the direction-indicating angel α . The new position (x, y, α) is calculated from the old one (x_0, y_0, α_0) using the values n_r and n_l given by the incremental encoders placed on each motor axis. The table below shows how the new position is calculated:

Table 1. Calculation of the change of position

				
$ n_l = n_r $	n_r S	n_r S	$n_r - n_l$ T	$n_r - n_l$ T
$ n_l < n_r $	n_l S	n_l S	$-2 * n_l$ T	$-2 * n_l$ T
	$n_r - n_l$ C	$n_r - n_l$ C	$n_r + n_l$ C	$n_r + n_l$ B
$ n_l > n_r $	n_r S	n_r S	$2 * n_r$ T	$2 * n_r$ T
	$n_r - n_l$ C	$n_r - n_l$ C	$-(n_r + n_l)$ C	$-(n_r + n_l)$ C

Strait: $x = x_0 + n \cdot \Delta l \cdot \cos \alpha$ $y = y_0 + n \cdot \Delta l \cdot \sin \alpha$	Turn: $\alpha = \alpha_0 + \frac{n}{2 \cdot r} \cdot \Delta l$	Curve: $\alpha = \alpha_0 + \frac{n}{2 \cdot r} \cdot \Delta l$ $x = x_0 + r \cdot (\cos \alpha - \cos \alpha_0)$ $y = y_0 + r \cdot (\sin \alpha - \sin \alpha_0)$
--------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4 Sensor Calibration

Adapting an one-dimensional self-organizing feature map [7, 8] by training with a real robot data set, we managed to calibrate the ambient light-sensors (see Fig.6), i.e. to transform the coarse grained information of Khepera's angle-to-light-source into a fine grained one; this also helped to overcome sensor-response variations due to fabrication tolerances. Calibration of the ambient light sensors means to classify the sensor vectors by an one-dimensional Kohonen feature map. Each neuron is active for a certain class of sensor vectors giving a direct information about Khepera's angle-to-light source.

The charts (Fig.7) show the sensor data, respectively used to learn and to test the neural network. Before usage, the data were standardized by a scaling process, which increased the accuracy considerably.

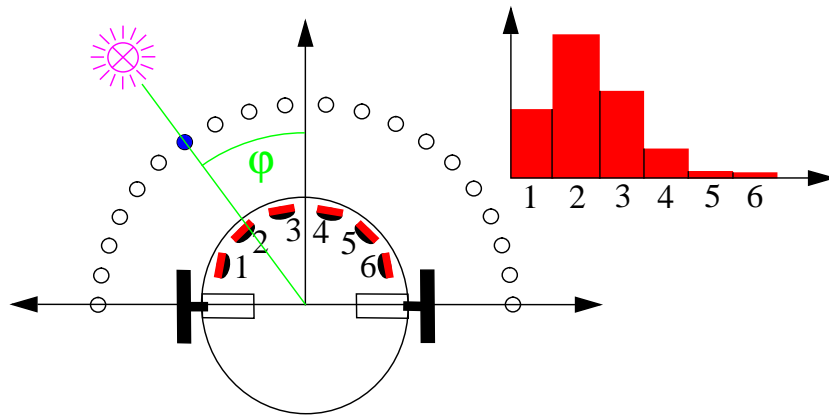


Fig. 6. The picture on the left shows that every neuron belongs to a specified angel. Each neuron is trained to react on a special characteristic of sensor values (e.g. the neuron represented by the filled circle reacts on the sensor vector given in the bar chart on the right).

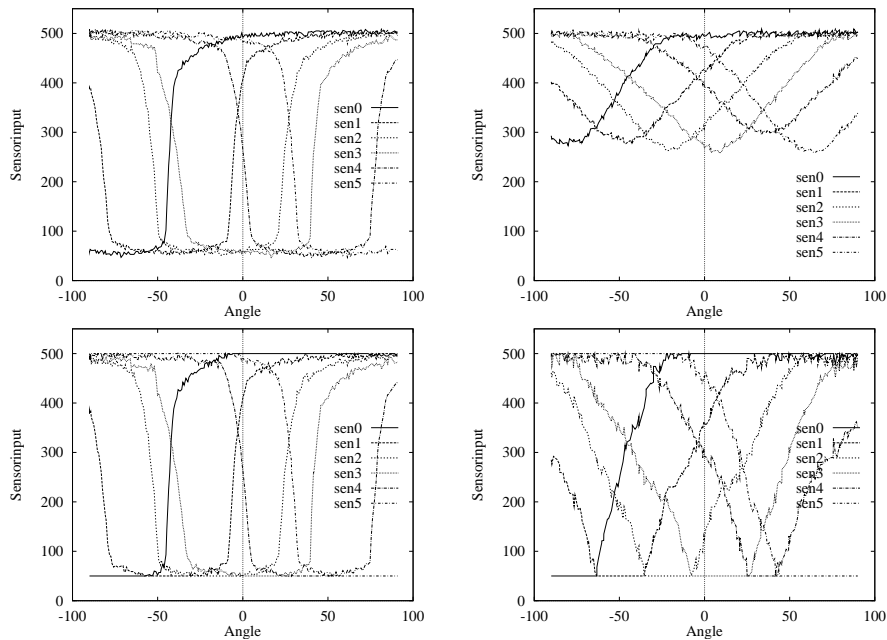


Fig. 7. The sensor data sets used to train and test the neural net; upper left: learning data, upper right: test data, lower left: scaled learning data, lower right: scaled test data.

The network consists of 60 neurons. It was trained in 20 cycles with an incremental change of the neighbourhood-width from 10 to 1 and a learning rate from 0.6 to 0.1. The result was an average error of 1.52 and a maximum one of 6.66 degrees at the recall process. The test at double distance caused an average error of 6.52 and a maximum one of 21.89 degrees. The results are graphically depicted in Fig.8.

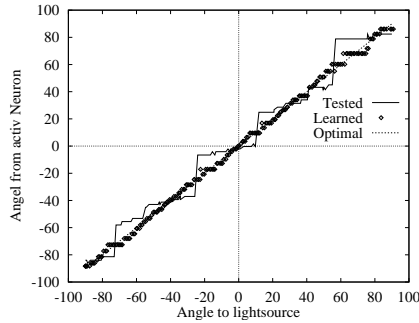


Fig. 8. The result of the adaptation process is shown in this chart. The dotted line serves as a referential, the bold dots correspond to the data being learned and the strait line belongs to data registered at the double distance with respect to the learning set.

5 Mapping Procedure

The procedure of the position-determination of detected light-sources was implemented as follows (see Fig.9). If the robot detects a light-source (1), Khepera approaches it (2) until one sensor value exceeds a certain threshold (optimal range for the neural network is reached) (3). Then Khepera registers its position vector (x_1, y_1, α_1) , turns by 90 degrees and moves in a strait line (4) to a second turning point (5). Afterwards, the robot reapproaches the light-source (6) until the optimal range is reached again (7). With the new position vector (x_2, y_2, α_2) and the old one, the location of the target can be calculated.

In this section, the conditions for changing the state of a light source are explained (see Fig.10). If Khepera penetrates the inner circle (in the Manhattan distance) of a previously discovered light source without detecting it, the state of the light source will be changed from on to off. If a “new” light source is registered within the outer circle of a previously discovered one, they are recognized as being identical. Eventually, the internal position of the robot is readjusted by a comparison of the previously and the currently registered location of the light source.

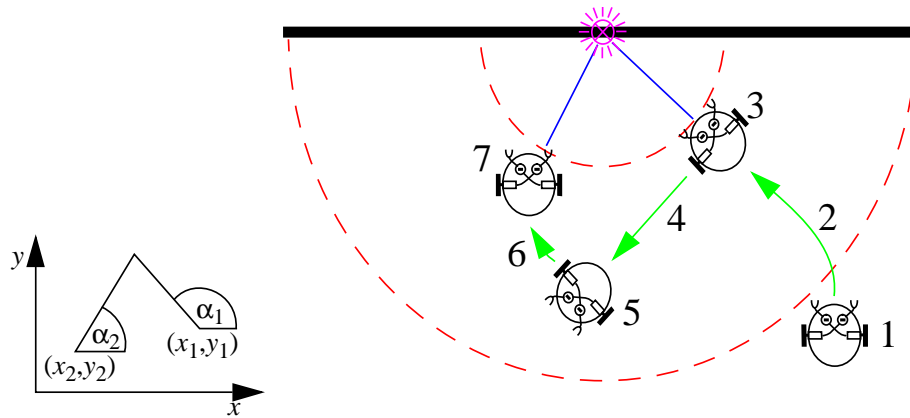


Fig. 9. A schematic view of the successive steps of the mapping procedure (right) and a sketch of the triangulation process using the registered data (left), hence determining the detected light source's location.

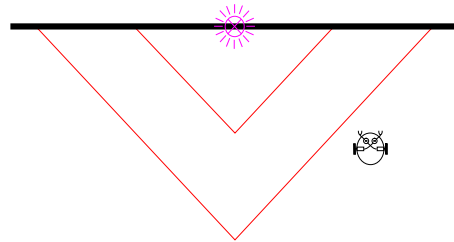


Fig. 10. Proportional view of Khepera and the two circles (in the Manhattan distance) relevant for changing the state of a previously discovered light source.

6 Report

We developed a simple code to enable the robot to report its findings during the last round using the two LEDs of Khepera.

The signals have the following meaning (legend: ● on, ○ off):

- ● New light source found
- ○ Previously discovered light source recognized
- ● Previously discovered light source turned off
- ● Previously discovered light source turned on again

7 Results

This paragraph is dedicated to the obtained results for both the simulator solution to the dynamical nightwatch's problem and the real-robot implementation (see Fig.11, Fig.12) using the obtained paths of the robot in the respective environments as an evaluation benchmark. We state a very satisfactory behaviour of the simulator solution whereas the real-robot version unveils to be feasible only locally. Note that the compiled C-code of the whole program has 87Kbyte, hence we are using only about one third of the RAM capacities of Khepera's processor.

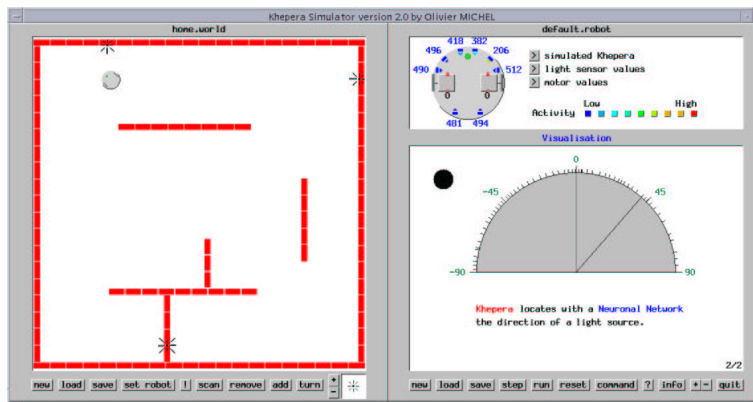


Fig. 11. In the visualisation window of the Khepera simulator [6], the robot's current angle-to-light-source is depicted.

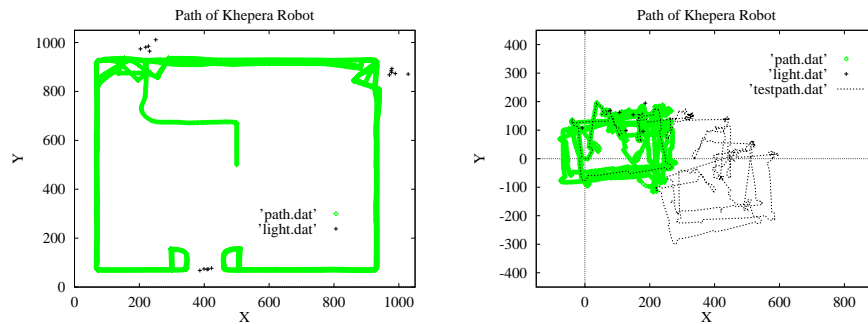


Fig. 12. The left gnuplot shows the path of the simulated Khepera. Note that the small location errors are solely caused by the remaining inaccuracies of the neural network. The right gnuplot shows the path of a real Khepera wherein the bold line represents the path with and the dotted line one without readjustment of the position using the already discovered light source as a landmark.

8 Conclusion and Future Work

Inspired by concepts of neural information processing and the agent paradigm, we implemented a solution to the dynamical nightwatch's problem, both on the simulator and on the real Khepera. In particular, we used an one-dimensional self-organizing feature map for sensor calibration attaining maximum accuracy in average. The simulator solution proofed to be globally reliable, whereas the real-robot implementation had some short-comings due to step-counting errors inducing inaccuracies into the position determination procedure. This was partly cured by taking into account the already registered light sources as local landmarks in order to readjust the current position. The results showed to be satisfactory locally, i.e. when either the robot's environment was restricted to a certain area or the runtime was limited.

Nevertheless, some further research on the topic of navigation is to be done. We hope to enhance the robot's performance considerably by incorporating associative memory and neural classifiers into the control structure. Moreover, it is also envisaged to replace some of the software modules by resource-efficient microelectronic devices as described for example in [9, 10].

References

1. S. Geva, J. Sitte and H. Sira-Ramirez, When are tasks "difficult" for learning controllers?, 1994 World Congress on Computational Intelligence WCCI, July 1994, Orlando, Florida, USA. Proceedings of the IEEE International Conference on Neural Networks, Volume IV, pp.2419-2423
2. P. Maes, Modeling Adaptive Autonomous Agents, Artificial Life Journal, edited by C. Langton, Vol 1, No. 1 & 2, MIT-Press, 1994
3. R. A. Brooks, Intelligence without Representation, Artificial Intelligence, 47, pp. 139-159, 1987
4. R. A. Brooks, Intelligence without Reason, Computers and Thought lecture, Proceedings of IJCAI 91, Sydney, Australia, 1991
5. V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press/Bradford Books, 1984
6. O. Michel, Khepera Simulator, version 2.0, 1996, <http://diwww.epfl.ch/lami/team/michel/khep-sim/>
7. K. Malmstrom, L. Munday, J. Sitte, A Simple Robust Robotic Vision System using Kohonen Feature Mapping, Proceedings of the 2nd IEEE Australia and New Zealand Conference on Intelligent Information Systems, pp. 135-139, 1994
8. T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, 2nd edition, 1988
9. A. Heitmann, J. Malin, C. Pintaske, U. Rückert, Digital VLSI Implementation of a Neural Associative Memory, 6th International Conference on Microelectronics for Neural Networks, Evolutionary & Fuzzy Systems, 24.-26. September, Dresden, Germany, 1997
10. S. Rüping, M. Porrman, U. Rückert, SOM Hardware-Accelerator, WSOM'97: Workshop on Self-Organizing Maps, June 4-6, pp. 136-141, Espoo, Finland, 1997