A. Löffler, J. Klahold, and U. Rückert. The Mini-Robot Khepera as a Foraging Animate: Synthesis and Analysis of Behaviour. In *Proceedings of the 5th International Heinz Nixdorf Symposium: Autonomous Minirobots for Research and Edutainment (AMiRE)*, volume 97 of *HNI-Verlagsschriftenreihe*, pp. 93–130, 2001.

The Mini-Robot Khepera as a Foraging Animate: Synthesis and Analysis of Behaviour

A. Löffler

System Dynamics and Simulation - ED514
Astrium GmbH, D-88039 Friedrichshafen

J. Klahold and U. Rückert

System and Circuit Technology, Heinz Nixdorf Institute,
Paderborn University, Fürstenallee 11, D-33102 Paderborn

E-mail: klahold@hni.uni-paderborn.de

Abstract. The work presented in this paper deals with the development of a methodology for resource-efficient behaviour synthesis on autonomous systems. In this context, a definition of a maximal problem with respect to the resources of a given system is introduced. It is elucidated by means of an exemplary implementation of the solution to such a problem using the mini-robot Khepera as the experimental platform. The described task consists of exploring an unknown and dynamically changing environment, collecting and transporting objects, which are associated with light-sources, and navigating to a home-base. The critical point is represented by the accumulated positioning errors in odometrical path-integration due to slippage. Therefore, adaptive sensor calibration using a specific variant of Kohonen's algorithm is applied in two cases to extract symbolic, e.g. geometric, information from the sub-symbolic sensor data, which is used to enhance position control by landmark mapping and orientation. In order to successfully handle the arising complex interactions, a heterogeneous control-architecture based on a parallel implementation of basic behaviours coupled by a rule-based central unit is proposed.

1 Introduction

When considering autonomous mobile robots, small systems are preferred to resolve partial problems applying a bottom up approach. They are relatively cheap and due to their small size the experimental environment stays small too. The restricted resources of such a system pose problems, which should be seen as a challenge and turned into an advantage. The limits regarding energy (period of validity), computing power, memory capacities, actuators (manipulators of environment), and sensors enforce the development of efficient solutions to the considered problems. For that reason, mostly partial aspects are considered up to now [2], e.g. obstacle avoidance [6], edge following, or simple navigation tasks [8]. The gained knowledge and results can be transferred onto large robots. However, the problem of integrating partial solutions to work together efficiently still remains.

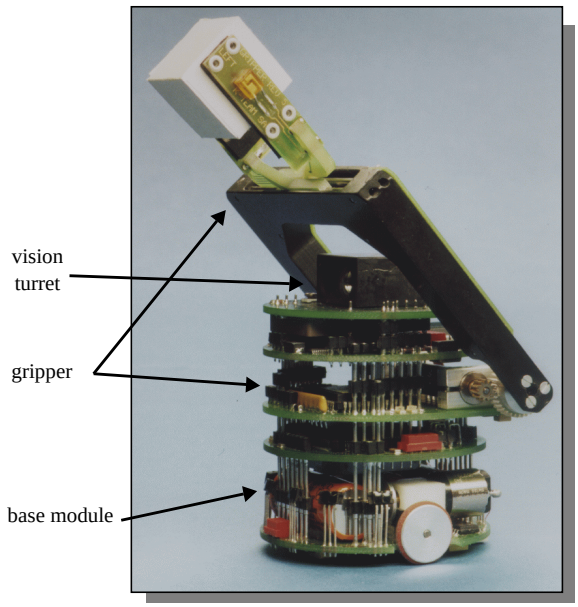


Figure 1. The mini-robot Khepera [14, 29] transporting a paper cube.

The focus of our research in the area of behaviour generation on autonomous systems follows the principle of maximal economy: The task may be defined as *to create maximal functionality with respect to a given system under the constraint of limited resources*. In the context of robotics, a problem may be considered as being (weakly) maximal, if its solution requires the use of all available sensor-resources and actuators of the system in question. The so called Dynamical Nightwatch's Problem presented in [21] by Löffler et. al. for example, does fulfil this criterion. This paper now describes substantial extensions to the work presented in the earlier article.

1.1 Presentation of the Experimental Platform

The experimental platform utilized is the mini-robot Khepera [14] with two additional turrets (Fig. 1). The base module has a height of 30 mm, a diameter of 55 mm, a mass of about 90 g and is equipped with a Motorola 68331 micro-controller (256 kByte RAM), two DC motors with incremental encoders, four Ni-Cd accumulators (4.8 V) for energy supply and eight infrared-proximity sensors (transmitters and receivers). The sensors may also be used for measuring the infrared intensity of the ambient light (receivers only) in which case they are referred to as light-sensors. Moreover, various add-on turrets are available, two of which are applied in the presented work: the gripper and the K213 vision turret (linear camera). The gripper may be moved to 256 positions and has two states (open/closed). The vision turret has an opening angle of $36°$ with a resolution of 64 pixels with a resolution of 8 bit; its maximal update frequency is 2 Hz. Due to the robot's small size and easy operability, it represents a versatile tool for scientific investigation in the field of autonomous systems. For a historical review concerning both the design and the development of the mini-robot Khepera as well as the underlying motivation, please refer to article [28] by Mondada et. al. in [23].

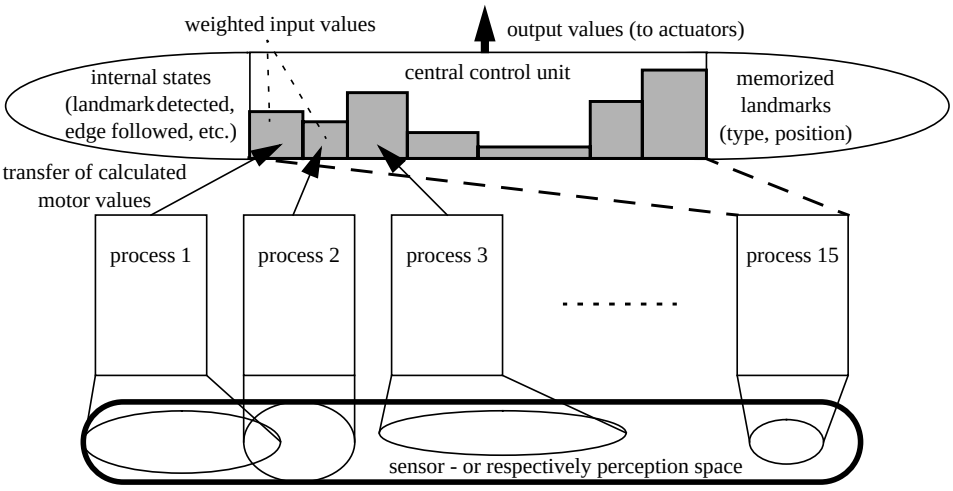


Figure 2. Schematic of the implementation concept, which assures the efficient usage of the given system resources.

1.2 Design Methodology Used

The task to implement a solution to a maximal problem has led to the development of a particularly adapted interaction concept (Fig. 2). The operating system of the Khepera allows the processing of up to 15 concurrent tasks. These tasks may represent the basic behavioural modules (e.g. obstacle avoidance, edge following, etc.) which operate on a subset of the available, multidimensional sensor space (8 infrared-/light-sensors, 2 incremental encoders, etc.). The motor values calculated by the respective basic modules are transferred to the central control unit that makes decisions about their (weighted) interaction due to both input-data (e.g. current sensor state) as well as internal states (e.g. memorized landmarks). The requests from the central module to transfer the calculated motor values occur in fixed time-intervals, which ensures the real-time capability of the system (if no new values are available, the old ones are taken). The update time is designed to maximize the number of program cycles which may be carried out during the autonomy time, which is determined by the allocated energy resources. Furthermore, the basic modules are designed in a simple and robust way, which does not overextend the sensor-hardware's range and precision and also guarantees fast data processing. Eventually, only the locations of meaningful objects, e.g. landmarks, are memorized which limits the required storage capacities. The observation of this concept assures a resource-efficient implementation. In this context, resource-efficiency may be defined as a Pareto-optimization problem: real-time capability has to be assured, the number of program cycles maximized (with respect to the available energy reserves) and the allocated memory minimized, whereby a correct accomplishment of the envisaged task is assumed. Note that one program cycle corresponds to one situation that the robot has to master. Therefore, the objective is to successfully cope with a maximum number of situations. The resulting overall optimization task represents a complex problem which cannot be solved analytically. Consequently, the illustrated interaction concept has to be accompanied by a particular design cycle (Fig. 3) which closely couples the behaviour synthesis, with various steps of analysis, leading to an iterative optimization of the program parameters. The behaviour synthesis is carried out using the bottom-up methodology, e.g. to succeedingly add new basic modules which entails the modification of the corresponding interaction schema (Fig. 2), and is constantly accompanied by simulations [26]. The applied methods are analysed by means of error propagation techniques whereby the sensor characteris-

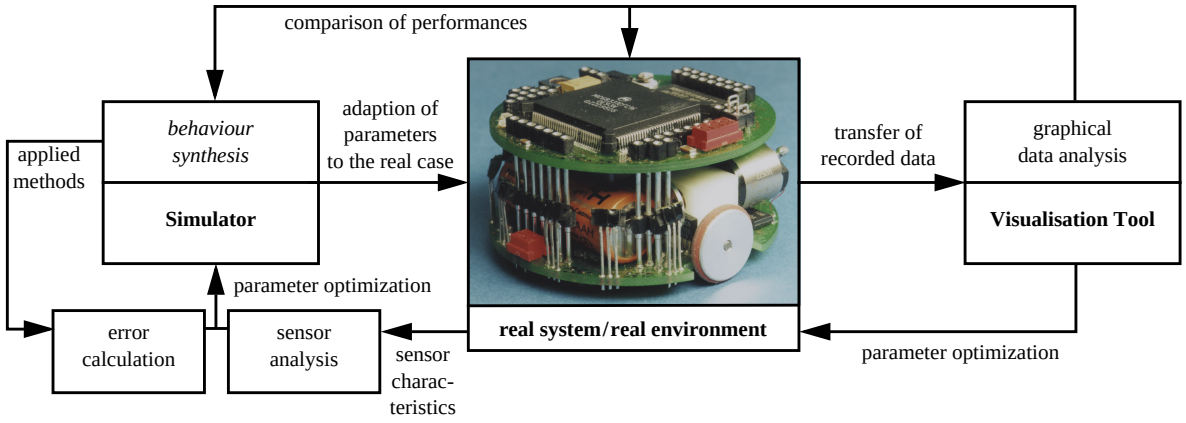


Figure 3. The design cycle. Essential to the proposed concept is the close coupling between behaviour synthesis and various analysis loops.

tics, on which the corresponding algorithms are based, are recorded with the real system. Both lead to an *a priori* parameter optimization. After having carried out the real experiment, the recorded data are graphically analysed using a specially developed visualization tool described in [19, 20]. This method may be applied to compare the performances of simulator solutions and real cases as well as to further optimize, i.e. in an *a posteriori* way, the program parameters.

The applied methods concerning the behaviour generating modules can be classified into three basic groups:
Firstly, geometrical calculations have been applied for position control tasks. They include both the odometrical path integration based on the incremental encoder values ('step counters') and the landmark orientation processes. This has been done for two reasons: on the one hand these problems are of geometrical nature and on the other hand they are responsive to calculus of error.
Secondly, Braitenberg [1] neural networks have been used to create simple, but robust basic behaviours like obstacle avoidance, edge following, etc. They have the advantage that, because of their reflexive nature, they do not require a high precision on the part of the sensor system. Moreover, they are not in conflict to the real-time demands due to their low complexity. Furthermore, a variant of Kohonen's algorithm [25] has been applied for sensor preprocessing in cases where accurate actions have to be carried out demanding a higher sensorial precision.
Thirdly, concerning the central unit which controls the interactions of the various modules, rule-based reasoning has been applied. This has the advantage that higher, i.e. 'cognitive-like', decisions may be carried out. Note that in cases where more than one module may contribute to current actions, no explicit subsumption architecture as proposed by Brooks [2] is used, but a weighted superposition of the incoming motor values, which leads to the establishment of an implicit priority system.

1.3 Organization of the Paper

This paper is organized as follows: after giving an overview of related works in section 2, sections 3 to 6 describe the three parts of the developed algorithms in detail:

- Firstly, the position control by means of odometrical path integrating (section 3) and reflexive schemata (section 4) are investigated. These schemata—exploration and navigation [22] algorithms—are based on the positioning system. Their particularity resides

in the fact that the environment is supposed to be unknown and not static, i.e. they are subjected to dynamical changes while the program is carried out.

- The second part describes a first approach to reduce the errors of the positioning system using additional sensor information. Therefore, section 5 gives an introduction to a methodology for adaptive sensor calibration [22], which allows to extract symbolic, i.e. geometrical, information from a sub-symbolic sensor data stream.

- The third part shows applications of the sensor calibration in two different cases (section 6). In the first one a local position correction is introduced and in the second, a global positioning system.

Eventually in section 7 the full-scale problem will be defined, the arising interactions discussed and the overall solution evaluated. The paper closes with a short conclusion (section 8).

2 Works Related to Our Approach

The related works discussed in this section have to be seen exemplary since they do not claim to be complete and should only be regarded in relation to this work. It is broadly divided into the three subsections—'simulated vs. real experiments', 'behaviour synthesis utilizing the mini-robot Khepera', 'navigation techniques'—and concludes with a short classification of the work presented in this paper.

2.1 Simulated vs. Real Experiments

De Sá et. al. [5] describes the (simulated) implementation of a controller comprising of three stages: the first level (reactive sensor-data processing) is realized by using a feedforward neural network, the second level (instinct-like manoeuvring) consists of pre-defined behaviour modules which are selected by a fuzzy controller, and the third level (cognitive motion planning) steers the global behaviour through rule-based reasoning.

Prem [32], on the one hand, extensively discusses the aspect of embodiment (embedding the controller into a real robot) which is especially crucial for hardware realisations of autonomous agents. On the other hand, Pipe et. al. [31] quantitatively analyze the learning performance of autonomous systems on the basis of simulations, concerning so-called cognitive maps under the assumption that senor and motor values are subjected to noise.

Miglino et. al. [27] presents results regarding the transfer of evolutionary developed controllers to real robots. This is particularly interesting in the context of genetic algorithms and similar techniques, because of the fact that generally a high number of evolution cycles have to be carried out to obtain viable results. This categorically forbids the use of a real system during the process of evolution. It is shown that an accurate model of the robot can be obtained, if real sensor characteristics are included, and that the introduction of conservative noise into the evolutionary process considerably smooths the transition to the real system, which—after carrying out some additional evolution cycles—shows the same performance as the simulated one.

Jakobi [13] contributes substantially to the ongoing discussion concerning the transfer of simulated controllers to real autonomous systems by developing the concept of the so-called minimal

simulation. When modelling system and environment, this approach proposes to exclusively concentrate on the most important parts of both. The concept is elucidated by an example with the mini-robot Khepera: a robot which drives towards a T-shaped crossway receives a light signal from one of the two sides and has to follow the corresponding way. The controller, which succeeded in this task, evolved through genetic algorithms (300 lines of commented C-code, 1000 generations, 4 h simulations on a Sun Ultra 1 workstation).

2.2 Behaviour Synthesis utilizing mini-robot Khepera

Lambrinos [18] proposes an interesting approach to on-line learning of a sense-of-orientation using an internal 'sun-compass' implemented by means of a two-dimensional Kohonen feature map. The task consists of obstacle avoidance combined with the return to the starting position after a given period of time, while orientating itself with respect to a globally visible light-source. The most frequently encountered error arises from a lateral drift with respect to the initial position.

Ziemke [39] is concerned with the design of a connectionist-architecture for action-selection on the basis of monolithic recurrent neural networks. The investigated (simulated) controller consists of two neural networks: a three-layer functional network (slave) that maps the sensor-input (first layer) to a state (second layer) and an output-vector (third layer) respectively, and a contextual network (master) which dynamically calculates the input weights of the functional network. The on-line adaption is done by backpropagation and typically requires about 15000 update cycles.

Floreano et. al. [7] presents an experiment with co-evolving predator-prey species (genetic algorithms with approximately 100 generations). The predator has an additional vision turret, but only drives with half of the prey's velocity. It is demonstrated that the prey utilizes sensor noise to generate random trajectories, whereas goal-directed following strategies are evolved by the predator.

Franz et. al. [9] describes a scene-matching technique using a so-called parametrical disparity field that is based on the comparison of a current visual impression with the memorized vicinity of the home-base, through which a homing behaviour can be realized. The average success rate (robot drives up to 1 cm to the goal within 30 sec, 20 test runs with random initial position) varies between 94 (distance to goal 5 cm) and 46 (25 cm) percent.

In the task of obstacle avoidance in a given environment, Floreano et. al. [6] compares evolutionary optimized controllers with a genetic algorithm (80 individuals, 20 genes each, 100 generations) to neural network solutions (three neurons with 27 synapses altogether). The results are comparable in their performance, whereas the neural networks show a faster convergence rate.

Scheier et. al. [35] investigates the constraints on neural controllers which arise through the special requirements of the autonomous system in question and their corresponding environment. It is shown that knowledge about the environmental structure (e.g. symmetries and the scaling of the obstacle structure) can be incorporated into the training algorithms leading to a reduction of their time-complexity.

Verschure et. al. [37] tries to reproduce the flexibility of biological life-forms by using a three-level (reactive, adaptive and cognitive) distributed controller. The described task consists of repeated navigation to different goals in an unknown environment. It is shown that adding the

second level reduces collisions with walls, but also the number of reached goals per time period (with respect to the one-level controller). In contrast, appending the third level leads to an increase in the number of encountered goals, while simultaneously decreasing collisions and reducing mileage (the calculation power of the system had to be extended by four PentiumPros and one Sun Ultra 1 workstation).

A comprehensive overview of current research projects in the area of behaviour implementation on the mini-robot Khepera can be found in the proceedings of the 1st International Khepera Workshop [23] covering the areas of 'artificial evolution', 'neural networks and learning', 'hardware design' and 'human interfaces'.

2.3 Navigation Techniques

Hoppen [11] defines the navigational task as the manoeuvring of a mobile object on the basis of incomplete information and the given restrictions to a pre-defined destination. In the case of an unknown environment, the so-called backtracking method is advocated: the system is moved on a straight line towards the envisaged destination; if an obstacle is encountered, the algorithm first avoids it and then tries to realign the robot. In case this re-alignment is not possible, the robot is driven back to the last branching point, where an alternative route is explored. Practical in most cases, this algorithm still runs short if the underlying position calculation is subjected to a high noise level.

Using the desert ant *cataglyphis* as biological model, Hartmann et. al. [10] presents a neural architecture which approximately implements a position control system on the basis of odometrical path integration. The proposed architecture consists of chain-like neural networks, where travelling activity patterns represent the path and angular differences. In the discussion section, it is outlined that other mechanisms, like landmark orientation, have to be added in order to correct the accumulated errors, e.g. due to slippage. The last point is particularly interesting with respect to the transfer of such position control mechanisms to mobile autonomous robots.

Icking et. al. [12] describes so-called competitive algorithms (the algorithmical solution is slower than the exact one only by a constant factor) for robot navigation. An intuitive example is the search for a door in a wall by alternately driving left and right while doubling the search range after each turn. Unfortunately, these kinds of algorithms are also of limited use for real robots, because they require high standards concerning both sensor range and precision.

Kurz [17] presents a method to create a topological map (a two dimensional Kohonen feature map with 10×10 neurons) of an unknown environment, which is based on a sensor fusion technique both applying ultrasonic and goniometry. The map is constructed using both the current ultrasonic data as well as a position calculated by path integration as input parameters, which requires a regular matrix of points of known positions used to correct position errors. Problems with this approach arose especially during experiments in cases where neither the structure of the environment nor the position of the robot were known.

Oore et. al. [30] proposes another interesting method where the robot is enabled to (in a quasi-unsupervised way) learn its location in non-static environments on the basis of noisy ultrasonic data. A probabilistic measure, which allows the prediction of the ultrasonic sensor data at a certain location by means of radial-basis-function networks, is introduced. On the other hand, previous ultrasonic data are used to correct the positioning system. The coupling of these two processes enables the system to learn a correlation between geometrical position and corresponding ultrasonic data. Moreover, the method shows good convergence characteristics under simulated conditions.

Tani et. al. [36] deals with self-organising processes to learn certain trajectories (e.g. 0- and 8-like ones). The proposed method is based on a two level neural structure. The first processing level, a three-dimensional Kohonen feature map ($6 \times 6 \times 6$ neurons with 24 synapses each), compresses the sensor data of a laser scanner. The second, a partly recurrent network (three levels with 5, 8, 3 neurons respectively), will only be activated at critical branching points where a binary decision is made. In real experiments, only a very low noise percentage could be tolerated without causing strong negative effects on the stability of the trajectories.

2.4 Classification of the Work Presented

To conclude this section, one may state that—in contrast to many previous publications [5, 30, 39]—the presented work does not limit itself to purely simulated systems and environments, but deliberately investigates the possibilities of behaviour generation on a real autonomous system, the mini-robot Khepera [28, 29], with its limited resources. Although several of the mentioned publications have dealt with the transfer from simulations to real experiments [13, 27, 31, 32], a new design methodology (cf. Fig. 2, 3) has to be applied to solve a maximal problem. Moreover, one may infer that—due to their differing research goals—previous experiments using the mini-robot Khepera as the implementation platform [6, 7, 37] only partly scooped out the attainable behavioural complexity of the system. Especially the question of a resource-efficient implementation [35] as well as—in a more general context—the analysis of methods applied and behaviour obtained [7, 17, 18, 36] have been treated only marginally. Eventually, the navigation techniques investigated are mostly limited to static or quasi-static environments [9, 10, 11, 12].

The following work now attempts to achieve maximal functionality with respect to the given resources of the system, wherein the common assumption of a static environment is omitted. Moreover, ample space is dedicated to the analysis of the methods applied and behaviour obtained. Note that the presented work follows the broad outline of the conceptual framework of the so-called ‘novelle artificial intelligence’ advocated for example by Brooks [3, 4] and Maes [24].

3 Position Control I

The exploration of an unknown environment and in particular the navigation towards a point-like destination requires the existence of a position control system. The mini-robot Khepera offers—through its incremental encoders—the possibility to implement such a position control system on the basis of odometrical path integration (Fig. 4). In the next two sections, a corresponding geometrical model is developed and critically analysed by classical error propagation techniques as well as graphical visualization.

3.1 Odometrical Path Integration

The current position (x_R, y_R, α_R) is calculated in an incremental way on the basis of the previous position $(x_{R_0}, y_{R_0}, \alpha_{R_0})$, the incremental encoder values (n_L, n_R), the wheel-distance $(d = 52\,\mathrm{mm})$ and the advancement per pulse $(\Delta l = 0.08\,\mathrm{mm})$. Two cases have to be differentiated. In the first one it is assumed, that the wheels of the robot are symmetrical driven (Eqn. 1-3). The remaining situations are handled by the second case (Eqn. 4-10).

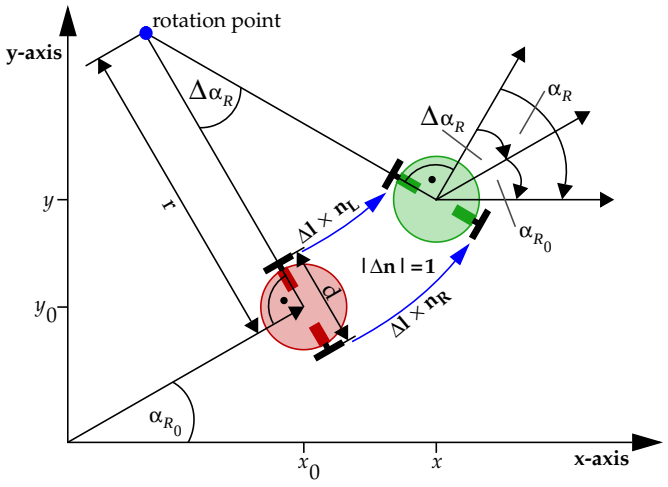


Figure 4. Graphical representation of the position calculation by path integration showing the current and the previous position.

1. case: $|n_L| = |n_R|$

$$\alpha_R = \alpha_{R_0} + (n_R - n_L)\,\frac{\Delta l}{d} \tag{1}$$

$$x_R = x_{R_0} + \frac{n_R + n_L}{2}\Delta l\cos(\alpha_R) \tag{2}$$

$$y_R = y_{R_0} + \frac{n_R + n_L}{2}\Delta l\sin(\alpha_R) \tag{3}$$

2. case: $|n_L| \neq |n_R|$

$$\Delta\alpha_R = \left|\left(|n_R| - \operatorname{sign}(n_R n_L)\,|n_L|\right)\frac{\Delta l}{d}\right| \tag{4}$$

$$r = \left|\left(\frac{|n_R| + |n_L|}{|n_R| - |n_L|}\right)^{\operatorname{sign}(n_R n_L)}\frac{d}{2}\right| \tag{5}$$

$$\Delta x_R = r - r\cos(\Delta\alpha_R) \tag{6}$$

$$\Delta y_R = r\sin(\Delta\alpha_R) \tag{7}$$

$$x_R = x_{R_0} + \operatorname{sign}(n_R + n_L)\Delta y_R\cos(\alpha_{R_0}) + \Delta x_R\cos\left(\alpha_{R_0} + \operatorname{sign}\left(|n_R| - |n_L|\right)\tfrac{\pi}{2}\right) \tag{8}$$

$$y_R = y_{R_0} + \operatorname{sign}(n_R + n_L)\Delta y_R\sin(\alpha_{R_0}) + \Delta x_R\sin\left(\alpha_{R_0} + \operatorname{sign}\left(|n_R| - |n_L|\right)\tfrac{\pi}{2}\right) \tag{9}$$

$$\alpha_R = \alpha_{R_0} + \Delta\alpha_R\operatorname{sign}\left(n_R - n_L\right) \tag{10}$$

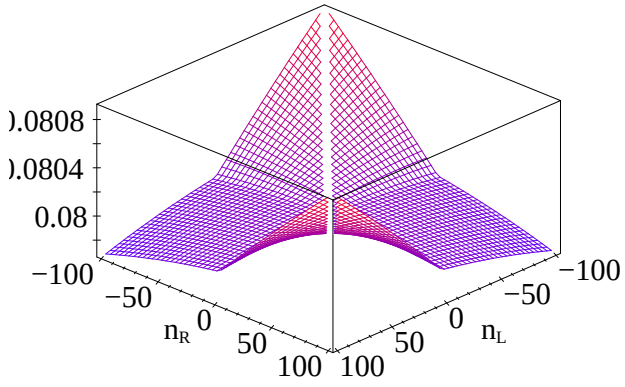


Figure 5. Arising position errors per program update cycle due to non-controllable influences, in particular slippage, during the process of odometrical path integration.

3.2 Error Propagation

In order to estimate the arising error, which mainly stems from slippage, the error in the incremental encoder values is assumed to be one ($F_{n_L} = F_{n_R} = 1$) per update cycle. They propagate as follows:

$$F_\alpha = \left|\frac{d\alpha_R}{dn_L}\right| F_{n_L} + \left|\frac{d\alpha_R}{dn_R}\right| F_{n_R} \tag{11}$$

$$F_x = \left|\frac{dx_R}{dn_L}\right| F_{n_L} + \left|\frac{dx_R}{dn_R}\right| F_{n_R} \tag{12}$$

$$F_y = \left|\frac{dy_R}{dn_L}\right| F_{n_L} + \left|\frac{dy_R}{dn_R}\right| F_{n_R} \tag{13}$$

$$F = \sqrt{{F_x}^2 + {F_y}^2} \tag{14}$$

Since the two functions F_x and F_y are orthogonal with respect to the angle α_R, the total error F (Fig. 5) does not depend on α_R; assuming that the incremental encoder values are smaller than 100 pulses, one obtains a maximal position error of 0.081 mm per update cycle. Moreover $F_\alpha = 0.18°$ is constant and in particular independent of the incremental encoder values. An example—corresponding to the exploration task described in section 4—may illustrate the effects of these errors: the position is calculated about 12.5 times per second in average and the program shall run for 80 sec which equals 1000 update cycles. A worst-case estimation leads to a maximal position error of 8.1 cm and a maximal angular error of $180°$. It can clearly be seen that the angular error is much more critical—we will return to this point in section 6.

3.3 Parameter Optimization

A crucial aspect is the choice of the parameters d and Δl. This may be alleviated by means of the developed visualization tool [19], which allows the testing of (and insofar optimize) different parameter pairs for an individual robot on the basis of only one recorded data set and the known geometry of the environment in question. An example may elucidate this point: in a real experiment, a Khepera followed the (internal) walls of a rectangular environment ($75 \times$

$\Delta l[\mathrm{mm}]$	0.076	0.077	0.078	0.079	**0.080**
$\Delta\alpha[^\circ]$	18.6	9.4	0.1	-9.1	**-18.3**
$\Delta l[\mathrm{mm}]$	0.081	0.082	0.083	0.084	
$\Delta\alpha[^\circ]$	-27.6	-36.8	-46.0	-55.2	

Table 1. Final angular error $\Delta\alpha$ with respect to variation of Δl (original configuration: bold).

$d[\mathrm{mm}]$	48	49	50	51	**52**
$\Delta\alpha[^\circ]$	-79.8	-63.5	-47.9	-32.81	**-18.3**
$d[\mathrm{mm}]$	53	54	55	56	
$\Delta\alpha[^\circ]$	-4.4	9.0	21.9	34.4	

Table 2. Final angular error $\Delta\alpha$ with respect to variation of d (original configuration: bold).

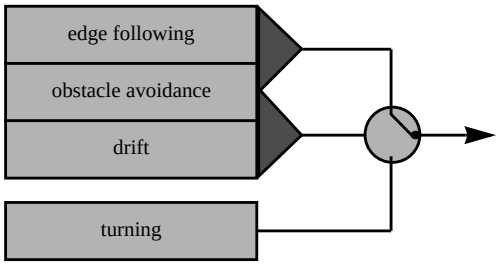


Figure 6. The interaction structure incorporating both the basic schemata responsible for the exploration of an unknown environment (without drift schema) as well as the point-to-point navigation to a destination of known position (including the drift schema).

$60\,\mathrm{cm}$) for roughly two rounds (i.e. 92 sec corresponding to 1126 program cycles) and stopped at an internal angle of 71.7° instead of the expected 90°. Tables 1 and 2 show the obtained final angular values with respect to variations of d and Δl respectively, where the original values ($d = 52\,\mathrm{mm}, \Delta l = 0.08\,\mathrm{mm}$) have been averaged over four different robots. From these data, one may conclude that the parameter pairs ($d = 52\,\mathrm{mm}, \Delta l = 0.078\,\mathrm{mm}$ or $d = 53\,\mathrm{mm}, \Delta l = 0.08\,\mathrm{mm}$ respectively) would have been better suited for the tested robot.

4 Exploration and Navigation

Exploration and navigation in unknown and potentially changing environments represent one of the basic and most intriguing problems concerning mobile autonomous robots. The most important constraints reside in the limited energy resources of such a system limiting its autonomy time, the real-time demands, the restricted calculation power, memory capacities and sensor range and precision. These severe constraints motivate the development of simple, but robust basic schemata with limited absolute time complexity. Moreover, the unknown and changing nature of the environment entails the use of heuristics as well as avoiding explicit maps for three reasons: the environment is non-static, the position control is subjected to considerable errors

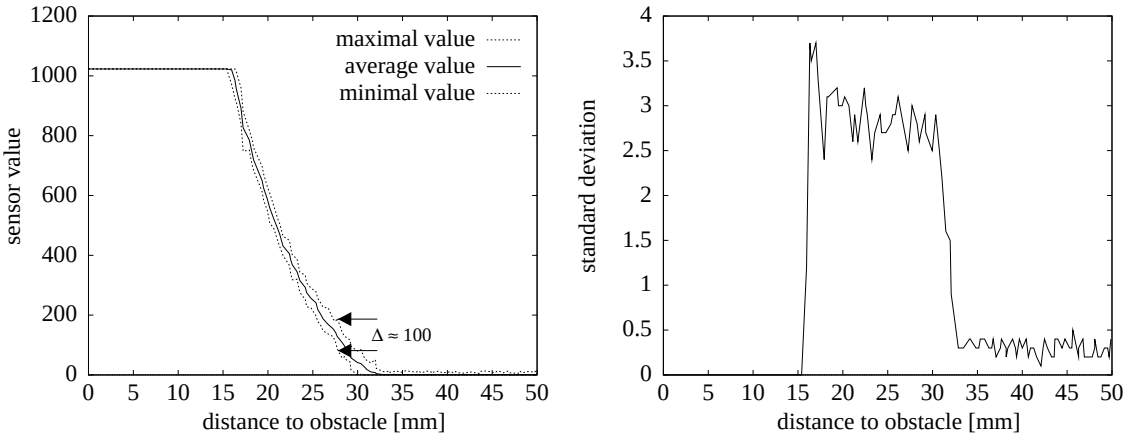


Figure 7. The distance characteristic (left) of the infrared sensors on the robot (sensor S_2, middle left) and the corresponding standard deviation (right), whereby the mean value has been taken from 50 samples.

and the available memory capacities are severely constricted. The exploration and navigation concept developed, incorporates the four basic modules of obstacle avoidance, edge following, drift and turning, whose interaction schema is depicted in Fig. 6. Thereby the motor output values of edge following (respectively drift) and obstacle avoidance are superimposed, wherein the obstacle avoidance has an implicit priority and also provides the basic propulsion. Eventually, in cases where turning on the spot is necessary, the schema turning suppresses the other ones. Before describing the developed algorithms, an *a priori* analysis of the infrared proximity sensors is presented, on which all of the algorithms in the current section rely.

4.1 Analysis of the Infrared Proximity Sensors

The recorded characteristics of the infrared proximity sensors and corresponding standard deviation (Fig. 7) show that the proximity sensitivity is limited to an interval from approximately 1.7 to 3.3 cm. Moreover, the maximal, respectively minimal values are shown, which may differ by $\Delta \approx 100$. For distances smaller than 1.7 cm the sensors saturate and above 3.3 cm only noise is detectable, which is caused by the quantization. Relating this to the system's diameter (5.5 cm), one realizes that the robot has to carry out its exploration task with a sensing system restricted to local perception. Moreover, despite the fact that the standard deviation with about 3 impulses seems to be quite low, the algorithms have to be designed in such a way that a noise level of 50 impulses can be tolerated.

Eventually, on the basis of the recorded sensor characteristics, a model for the visualization tool was developed (Fig. 8 left). The scalar sensor values are transformed to areas in which an obstacle may be located (Fig. 8 right). The thickness of the characteristics represents the corresponding noise level. This sensor model may be utilized for the process of *a posteriori* graphical analysis, thereby reconstructing the system's perception of its environment (Fig. 9). The reconstruction consists of two steps; first the accumulated sensor characteristics are drawn (Fig. 9 middle), and then the parts the robot travelled across are erased to correct the view (Fig. 9 right). The real environment (Fig. 9 left) is given as reference.

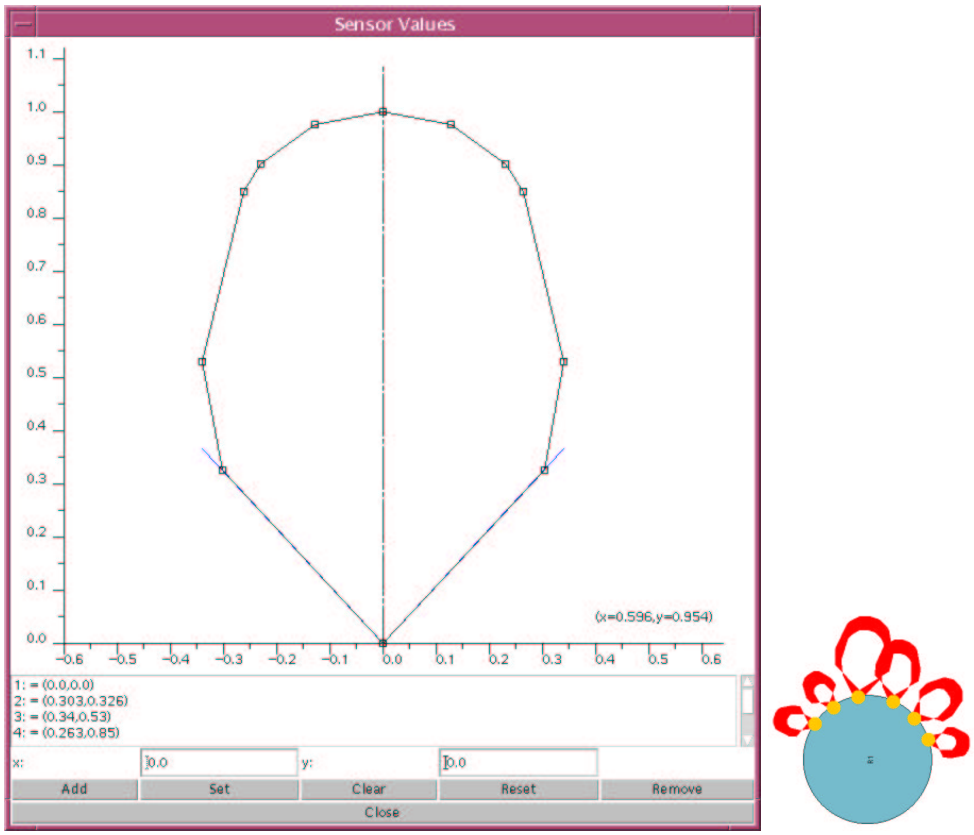


Figure 8. The scalar sensor values are—by means of the adjustable angular characteristics (left)—transformed to areas in which an obstacle may be located (right).

4.2 Obstacle Avoidance

Obstacle avoidance is one of the most basic tasks for mobile autonomous robotics. The presented algorithm is inspired by Braitenberg's vehicle IIIb [1] and also provides the basic propulsion. The velocity reduction of the right motor due to an obstacle on the left (Fig. 10 left) is represented by the dashed arrow. In order to get the relationship betweendistance-to-obstacle and speed, The activation function of the neurons (Fig. 10 right) has to be superimposed on the sensor characteristic (Fig. 7). The right (respectively the left) front sensors' values and the corresponding weight vector ($\mathrm{w}_0 = \mathrm{w}_5 = 0$, $\mathrm{w}_1 \ldots \mathrm{w}_4 = 1/2$) are multiplied by means of the dot product and transferred in an inhibitory way to the opposite motor. The initialization of the weights may be considered as a sensitivity function; the fact $\mathrm{w}_0 = \mathrm{w}_5 = 0$ is necessary to guarantee a smooth interaction with the edge following algorithm (see below). Moreover, the activation function of the two neurons represents a modified step function (the threshold is randomly chosen to be 475 ± 25, which avoids the formation of a preference direction) combined with a proportionality relation, permitting a smoother driving behaviour.

4.3 Edge Following

The edge following algorithm enables the system to leave room-like substructures of the environment in a time-efficient way. The basic concept consists of keeping the distance-to-wall

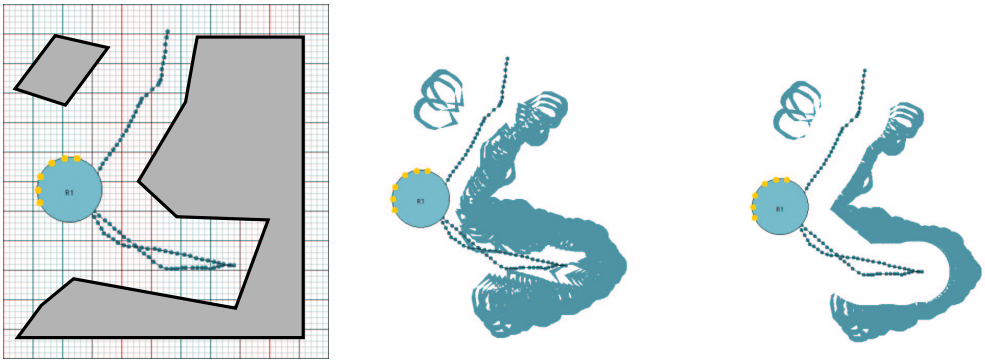


Figure 9. On the basis of the sensor model it is possible to regain a notion of the perceived environmental structure.

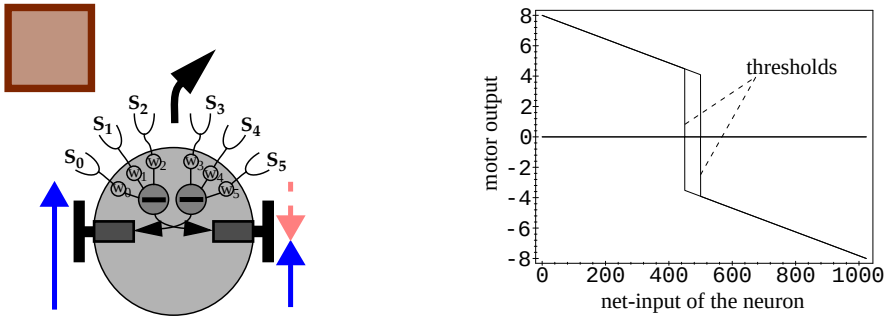


Figure 10. Representation of the neural network that implements the obstacle avoidance behaviour (left) with the activation function of the neurons (right).

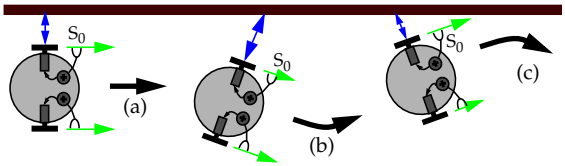


Figure 11. Behaviour implemented during the edge following algorithm.

constant, i.e. to implement a control system which keeps the sensor value S_0 (resp. S_5) constant. Thereby three different situations may occur (Fig. 11):

a) driving along a newly found wall (the sensor value S_0 will be memorized);

b) approaching a wall if the robot has drifted away (current sensor value is lower than the memorized one);

c) the robot drives away from the wall by means of the obstacle avoidance schema.

Moreover, a neural interpretation of the edge following algorithm is shown in Fig. 12. Therein a neuron, represented by a circle, calculates the sum of its input values, represented by incoming arrows, if not specified otherwise. If the difference of the current to the previous sensor value is smaller than a noise discriminating threshold of -7 and is detected, a drift value of -1 is

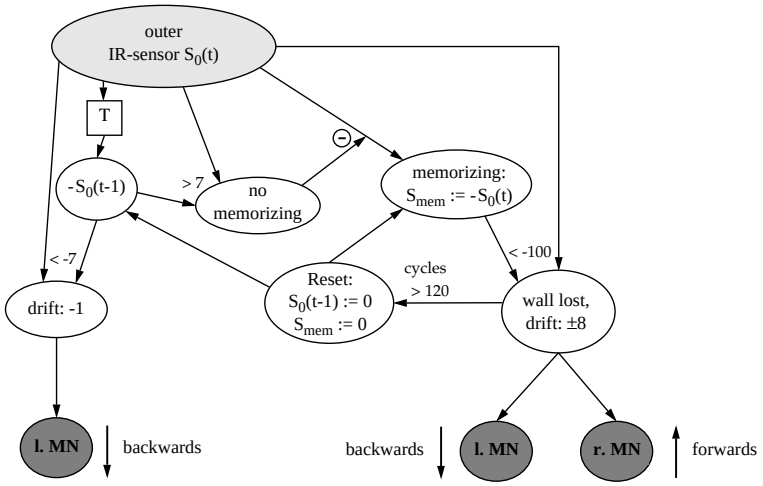


Figure 12. Neural interpretation of the edge following algorithm (the wall is assumed to be on the left side of the robot; l./r. MN ≙ left-/right motor neuron).

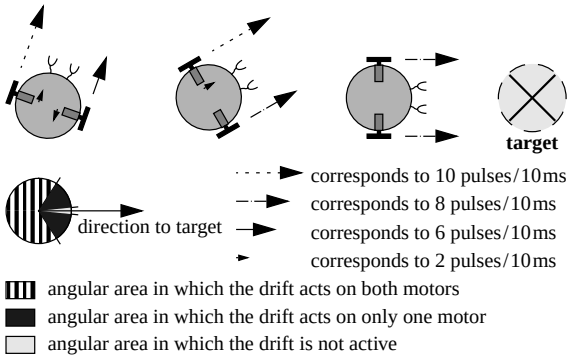


Figure 13. Representation of the effects of the drift schema depending on the angular miss-alignment.

superimposed on the left motor. If the difference is greater than 7 (approximately twice the standard deviation), i.e. the robot is approaching the wall, the current sensor value will be memorized ($S_{mem} = -S_0(t)$). In case the robot loses the wall ($S_0(t) + S_{mem} < -100$), a strong curve towards the wall will be driven until either the wall is relocated ($S_{mem} = -S_0(t)$) or 120 program cycles have been carried out. The edge following process starts with a reset.

4.4 Drift

The drift schema in conjunction with the obstacle avoidance and the turning algorithm (which may be considered as a special case of the drift schema) enables the robot to navigate to a point of known coordinates. Note that—for the discussed reasons—no map of the environment is available, e.g. no optimal path may be calculated; in consequence, this schema tries to manoeuvre the robot towards its destination on a straight line. An initial turn on the spot (Fig. 14 (1)) orientates the robot towards its navigational destination. If, during the process of manoeuvring towards this destination an obstacle is encountered (2), it will be avoided and the system will try to reorientate itself (3) by superimposing an adequate drift with respect to the angular miss-alignment (Fig. 13). Moreover, in order to avoid cyclic movements, the nearest point reached is

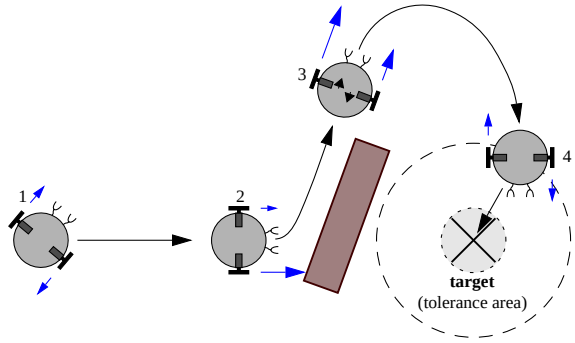


Figure 14. Example of a navigation sequence.

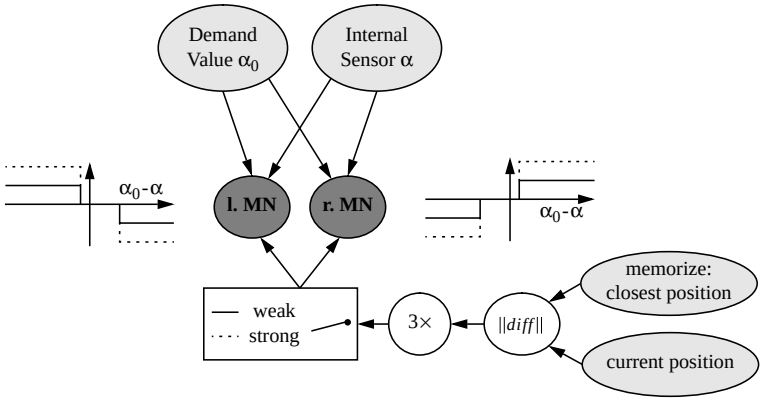


Figure 15. Neural structure that implements the drift (Fig. 13) as well as the turning schemata (Fig. 16).

memorized; in case this point is reencountered for the second time (with a tolerance of 5 cm), the drift is halved which leads to wider movements permitting the robot to leave small environmental substructures. If this still leads to cyclic behaviour, the navigation algorithm will be interrupted for a fixed (but increasing with the number of failures) number of program cycles in order to search a new starting point, which in practice amounts to an edge following behaviour. Finally, if the robot has approached its target by 15 cm (Fig. 14 (4)), a turn on the spot to reorientate the system is carried out and the basic propulsion is halved, avoiding an 'overshot'. In case the target is reached within 5 cm, the task is considered to be completed. This tolerance area concerning both target and nearest-point-reached is introduced due to the errors stemming from the path integration process. Eventually, the drift schema may also be interpreted in a neural way (Fig. 15).

4.5 Turning

The presented structure for implementing the drift schema (Fig. 15) may also be utilized to realize a turning-on-the-spot algorithm, where only the thresholds as well as the switch between a strong and a weak drift cease to exist. The resulting behaviour together with a visualization for two criteria causing a turn of $180°$ are shown in (Fig. 16).

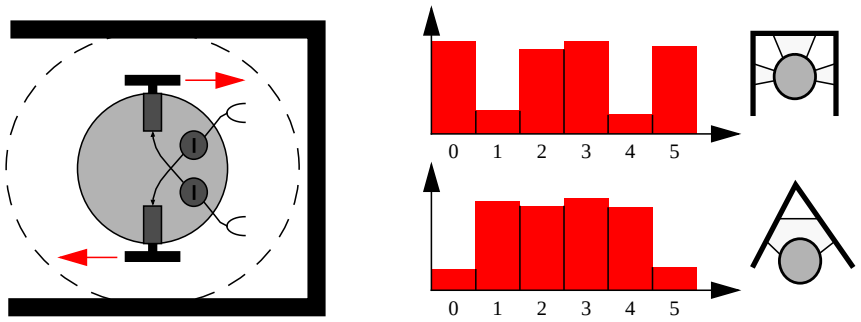


Figure 16. If it is not possible to leave a pre-defined circle during a given number of program cycles (left) or if its front IR-sensors perceive certain characteristical patterns (right), the robot performs a $180°$ turn.

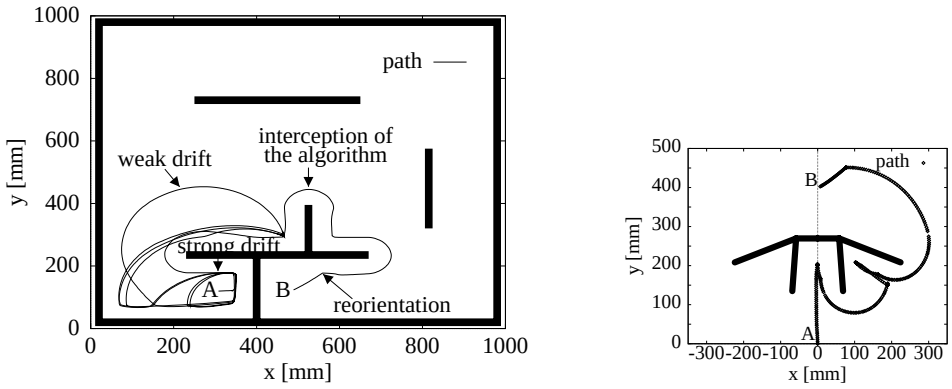


Figure 17. Trajectory of the robot during the A-to-B-navigation process in a simulated (left) and a real (right) environment.

4.6 Results: Point-to-Point Navigation

Concerning the simulator (Fig. 17 left), the developed navigation algorithm shows a very good behaviour in case of 'open' environments, i.e. structures without narrow, maze-like obstacle arrangements, for which the chosen parameter set is not adapted. This example (Fig. 17 left) points out the different drift factors (strong/weak) as well as the interception of the algorithm and the reorientation in the vicinity of the target. In the real case (Fig. 17 right), a comparable performance may be achieved on a smaller scale. Although a particularly harsh structure (two successive dead ends) was chosen, the robot reached the envisaged target with a deviation of $\Delta x = 2\,\mathrm{cm}$ and $\Delta y = 1\,\mathrm{cm}$ which is well inside the tolerance area.

It is nevertheless obvious that the proposed navigation algorithm is heavily dependent on the underlying position calculation. For this reason, the next two sections will see the introduction of a technique for adaptive sensor calibration and its application to enhance the positioning system.

5 Adaptive Sensor Calibration

Adaptive sensor calibration provides a means by which two different goals may be achieved: on the one hand, it allows the extraction of symbolic information from the generally sub-symbolic

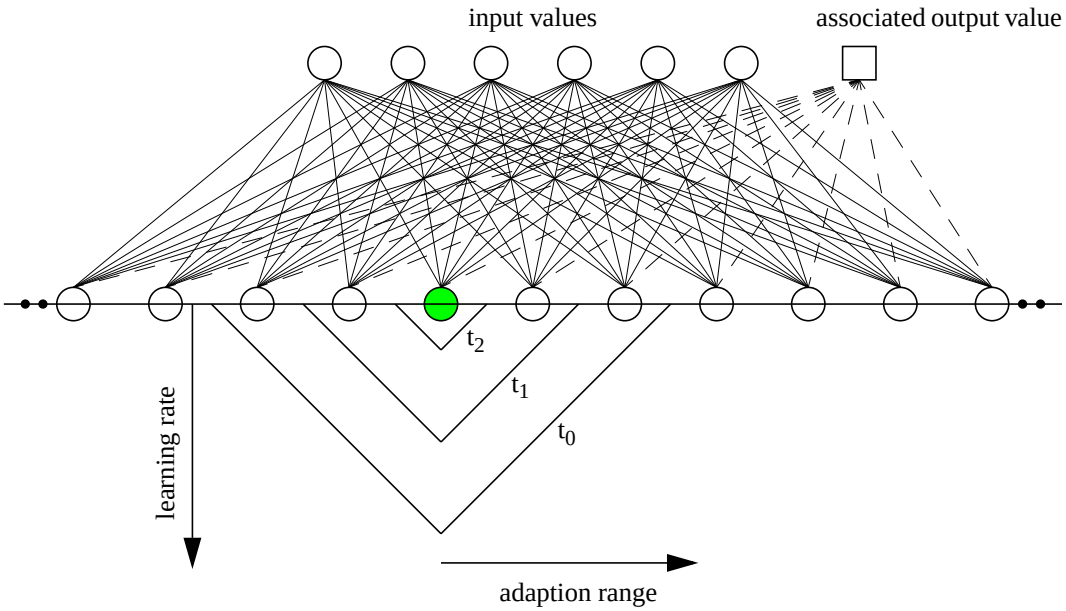


Figure 18. Schematic of a 1-dimensional self organising map (SOM) during the training period.

sensor data, and on the other hand, the quite intriguing problem of varying responses of individual sensors to identical circumstances, due to fabrication tolerances, may be approached. In the context of the presented work, a methodology on the basis of a variant of Kohonen's algorithm [16] inspired by Malmstroem et. al. [25] is used to extract geometrical informations from high dimensional sensor data in two cases: the calibration of the light-sensors leads to an angle-to-light-source value and the calibration of the linear camera provides distance-to-landmark information. First of all, the underlying algorithm shall be briefly explained.

5.1 Description of the Algorithm

The basic idea consists of approximating a function f which maps an n-dimensional input vector onto an m-dimensional output vector by classifying the input vectors, wherein a particular output vector is associated with every class. In the following, the output will be a scalar value due to the particular structure of the encountered problems; please note however, that the presented methodology is not restricted to this case.

$$\mathrm{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^m \quad \text{, here m = 1} \tag{15}$$

The methodology is divided into a training and a recall phase. In the training phase, it is assumed that a suitable data set is available (either by previous recording or by on-line input) which consists of example vectors containing both the sensorial data ($\vec{s}$) as well as the associated output value (o). The network consists of k randomly initialised neurons which match the example vectors regarding their dimensions. Moreover, they have fixed locations, i.e. it is assumed that the vector i is adjacent to the vectors $i-1$ and $i+1$, which results in a 1-dimensional structure being the most appropriate for the investigated cases. Again this does not represent a principal restriction; on the contrary, the dimension of the network should be adapted to the dimension of the output vector in question, which considerably simplifies the visualization of the results.

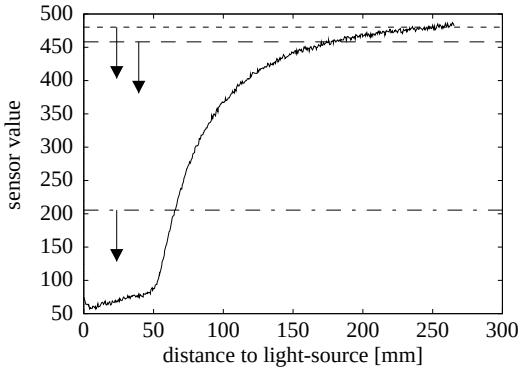


Figure 19. Characteristic curve of the light-sensors.

During the training phase, the neuron with the minimal distance to the current example vector (Eqn. 16) is determined (best match) and all neurons (Eqn. 17) are adapted towards the example (Eqn. 18, 19), whereby the adaption strength λ is dependent on the distance of the neuron in question to the best match. Moreover, the adaption strength decreases in time which leads to a quick but coarse ordering in an early phase of the training process and fine-tuning later on. In the given example (Fig. 18) the triangles represent the various learning rates and adaption ranges at different times ($t_0 < t_1 < t_2$). The marked neuron represents the best match, which means the neuron with the weight vector closest to the input vector. The vectors contain the input values and the corresponding output value associated by training. The training phase ends when no further decrease of the classification error can be detected. Note that the number of neurons utilized is dependent on the requirements of the application in question.

$$\vec{V}_{ex} = (s_0 \ldots s_n, o) \tag{16}$$

$$\vec{V}_i = (\vec{w}_i, \varphi_i) \tag{17}$$

$$\min_{i=1\ldots k} \left\| \vec{V}_i - \vec{V}_{ex} \right\| \quad \Rightarrow \quad \vec{V}_{bm} \tag{18}$$

$$\vec{V}_i(t+1) = \vec{V}_i(t) + \lambda\,(t, |i - bm|)\,\vec{V}_{bm}(t) \tag{19}$$

During the recall phase, only the sensorial data (Eqn. 20) is available to the system; consequently, the best-match search is carried out considering only the first n components of the network vectors. The complementary part associated with this best-match represents the required output (Eqn. 21), i.e. the complete n+m-dimensional information is recovered by means of a classification restricted to a n-dimensional subspace.

$$\vec{s}_{re} = (s_0 \ldots s_n) \tag{20}$$

$$\min_{i=1\ldots k} \|\vec{w}_i - \vec{s}_{re}\| \quad \Rightarrow \quad \vec{w}_{bm} \quad \Rightarrow \quad \mathrm{f}(\vec{s}_{re}) = \varphi_{bm} \tag{21}$$

5.2 Analysis of the Light-Sensors

The light-sensors (passively) measure the intensity of the infrared part of the ambient light. A characteristic, which has been obtained by a single measurement, is shown in (Fig. 19); the sensor value decreases with increasing intensity, i.e. decreasing distance to the light-source. When using a light-source with different intensity, the curve will be spread or compressed. In

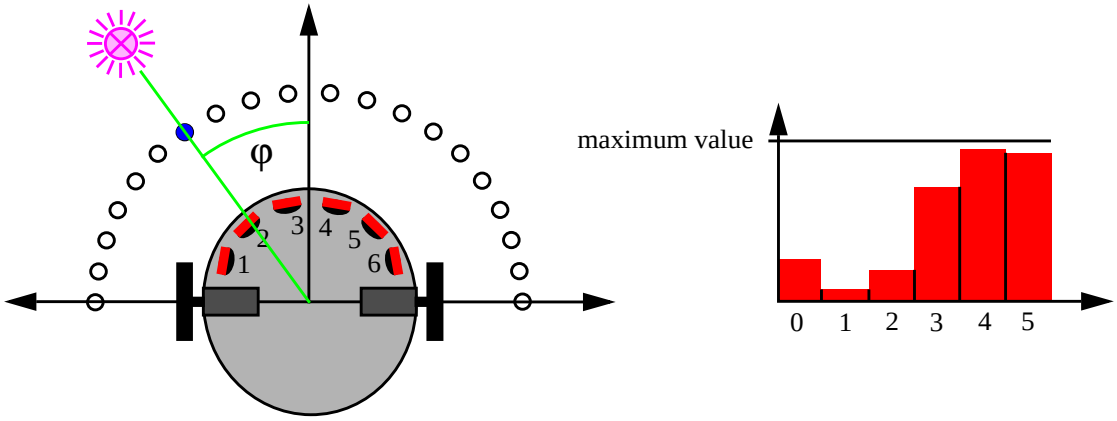


Figure 20. Schematic of a 1-dimensional self-organizing neural network which is used to extract a symbolic angle-to-light-source information (left) from the sub-symbolic sensor stream (right). Note that the sensor values are inverted (Fig. 19).

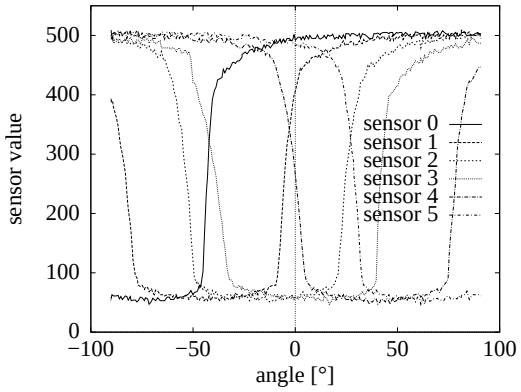


Figure 21. Set of sensor values used for training the neural network.

this context, it should be noted that the measured characteristic depends on the intensity of the light-source in relation to the background infrared radiation, which prompts one to use the light-sensors in a way which is independent of absolute intensity values. Furthermore, one may artificially create 'events', such as for example 'light-source encountered', by introducing corresponding intensity thresholds (dashed lines in Fig. 19), which have to be initially adapted to the background radiation. Note that noise plays a significantly lesser role here as for the (active) proximity sensors.

5.3 Calibration of the Light-Sensors

Fig. 20 shows in a schematic way the task of the light-sensor network which consists of transforming the 6-dimensional intensity signal to 1-dimensional angular information. For the training phase, a data set is obtained by placing a robot at a distance of 1 cm (4 cm for the test data set) to a light-source and rotating it by $180°$ while recording both sensor vectors (Fig. 21) and the associated angle. The latter has been obtained by odometrical path integration on the basis of the incremental encoders. In order to take the significant distance dependence into account, a scaling operation is performed (Fig. 22) prior to training and recall. This operation scales the

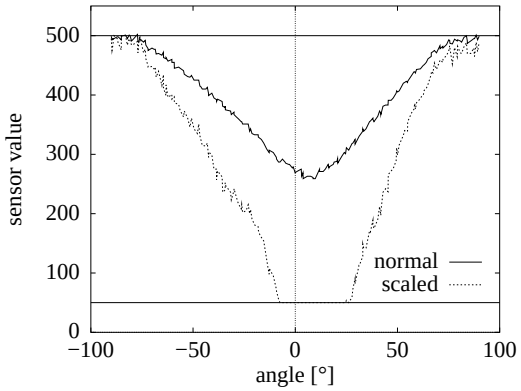


Figure 22. Set of values from sensor S_3 in a quadruplicate distance to the light-source with respect to the one the SOM was trained with.

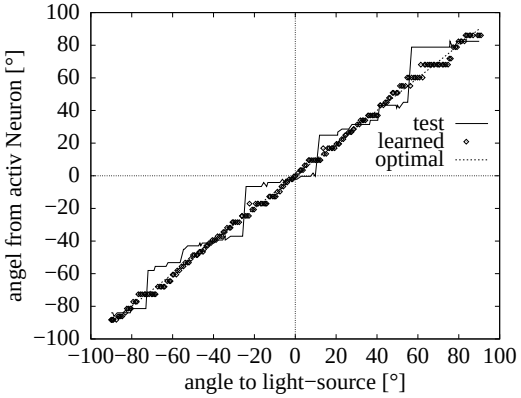


Figure 23. Result of the training, showing the test of the neural network with the data set used for learning (fat dotted line) and respectively with a data set resulting from a quadruplicate distance to the light-source (thin solid line). As reference a straight line is drawn. In spite of the scaling, a large distance sensitivity may be recognized.

sensor values, so that one of the six values has the normed lowest amount (50) and another one has the normed highest amount (500). This is more advantageous than performing a normalisation because of the border effects induced by the non-isotropic sensor arrangement (Fig. 20 left).

The chosen network consists of 60 neurons and has been adapted in 20 cycles while reducing the adaption width from 10 to 1 and the learning rate from 0.6 to 0.1 (Fig. 23). The average error amounts to the expected 1.52 degrees (the 60 neurons are distributed over the $180°$ with a spacing of $3°$). In contrast, the maximum error of $6.52°$, occurring at the borders due to the non-isotropic sensor arrangement. Despite the scaling operation, a strong distance sensitivity remains, which allows one to use the obtained network only at an optimal 'working' intensity. Using the visualization tool the current output of the network can be eventually displayed (Fig. 24).

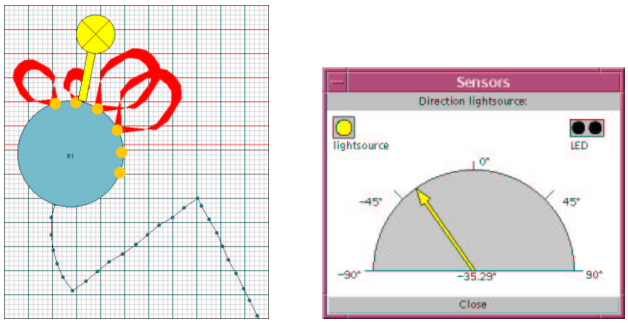


Figure 24. Visualization of the trained neural network for the light-sensor calibration.

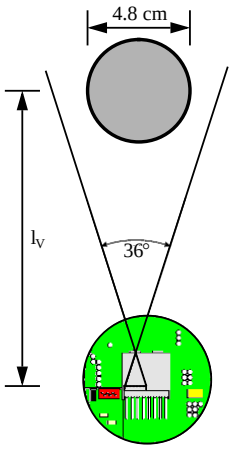


Figure 25. Schematic illustrating the recognition of a cylinder as a landmark using the linear camera.

5.4 Vision Data Preprocessing

The calibration of the linear camera provides distance information with respect to a particular cylindrical landmark (Fig. 25). In order to reduce the data dimensionality (64 pixel with a resolution of 8 bit), a preprocessing is carried out. Firstly, a gradient $\mathrm{G_i}$ of the original image $\mathrm{B_i}$ is calculated (Eqn. 22) and subjected to a noise suppressing threshold (Eqn. 23), which allows the detection of the right and left edges of the object, thereby permitting the identification of the landmark.

$$G_i = B_i - B_{i+1} \tag{22}$$

$$|G_i| < 20 \quad \rightarrow \quad G_i = 0 \tag{23}$$

The result of this operation on a typical image is shown in (Fig. 26). Every image is the average of 10 recordings; if in at least 4 recordings the system fails to identify two edges, the process is stopped. Moreover, Fig. 27 shows that next to the width of the perceived object, the minimal pixel value also carries some distance information, whereas the gradient values at the edges are subjected to too much noise and the maximal value saturates at small distances. Therefore, width and minimal values are chosen to be the sensorial input variables. Note that due to the diameter of the utilized landmarks (about 4.8 cm) and the opening angle of the camera $(36°)$, a minimal operating distance of 10 cm is necessary; above 40 cm, the resolution of the camera does not permit a viable operation either.

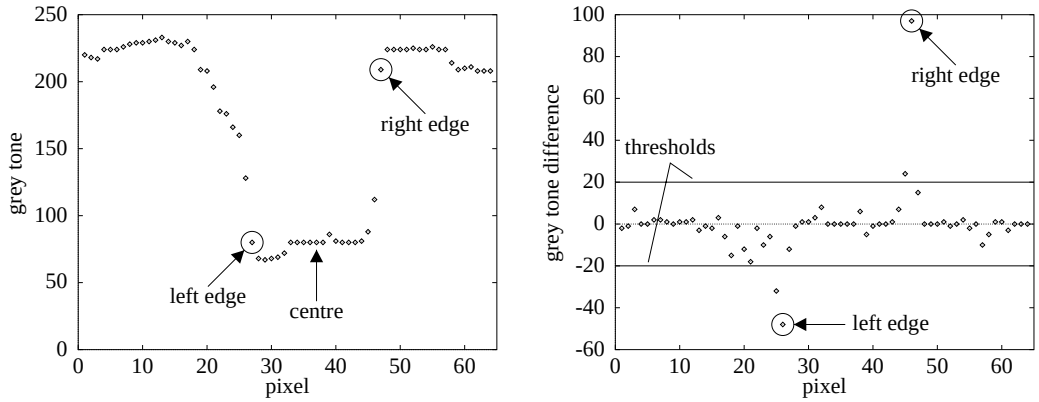


Figure 26. Picture of a landmark in view of the robot as sub-symbolic vision data (left) and the resulting gradient picture (right).

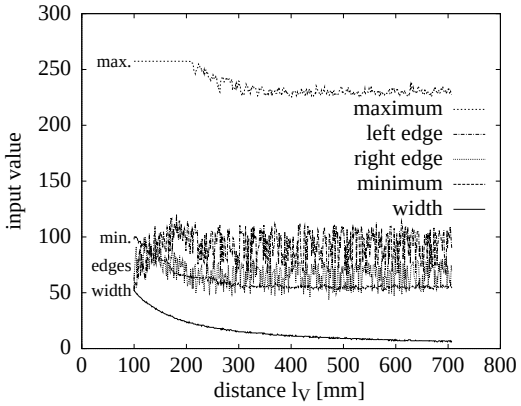


Figure 27. Diagram of several characteristics of the sub-symbolic vision data.

5.5 Calibration of the Linear Camera

Due to the fact that only two input components and one output value are present in this example, the calibration of the linear camera appears suitable for a comparison between off-line and on-line training. A network consisting of 100 neurons has been trained in both cases with only three cycles decreasing the adaption width from 10 to 1 and the learning rate from 0.8 to 0.1. The on-line training required an initialization with the recorded data from the first training cycle (Fig. 28). The result has been an average error of 5 mm (on-line: 9 mm) and a maximal one of 24 mm (48 mm). The significantly lower performance in the on-line case is due to odometrical errors which stem from a very discontinuous (periodic ac- and decelerations) movement imposed on the system by the calculation time needed for vision data processing and weight adaption. Moreover, in the close range of 10 to 20 cm, an expected average error of 1.5 mm has been obtained in the off-line case by distributing 100 neurons over 30 cm.

6 Position Control II

The two symbolic values obtained by adaptive sensor calibration, e.g. angle-to-light-source and distance-to-landmark, shall be used in the following to enhance the positioning system. This

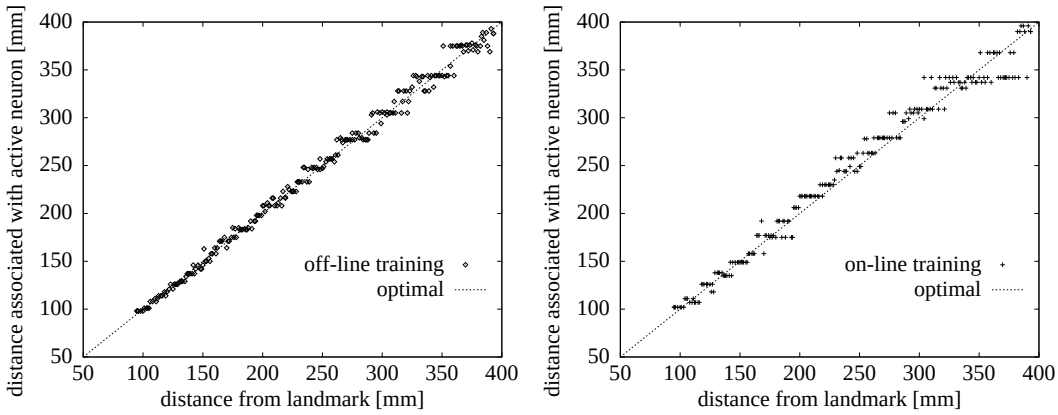


Figure 28. The diagrams show the results of the off-line (on a workstation, left) and on-line training (on the robots processor, right).

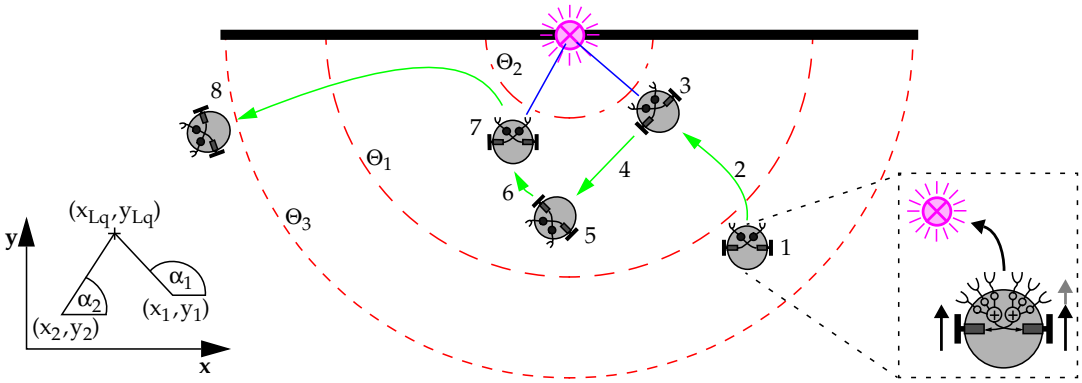


Figure 29. Mapping sequence of a light-source. The areas marked by the thresholds Θ_1-Θ_2 correspond to the dashed lines in Fig. 19.

is done in two consecutive steps: firstly, the angular information is used to map the locations of encountered light-sources by means of a triangulation process—if this light-source is recognized again in the following, the difference between its current and the memorized position is used to update the internal position of the robot itself (assuming that the light-source has not moved). Secondly, the distance information is applied to map the positions of three, globally visible landmarks during an initialization phase; by measuring (in an event-driven way) the angular differences under which these three landmarks appear, this knowledge may be used in the following to update not only the internal position of the robot, but also its orientation.

6.1 Cartography of the Light-Sources

A typical mapping sequence is shown in Fig. 29. If the robot detects a light-source (1, $\Theta_1 = 470$), it drives in this direction (2) until the average sensor value drops below a certain threshold $\Theta_2 = 200$ which signals that an optimal distance to the light-source (3) is reached. This minimizes the error of the neural map. After the robot's position at point 3 (x_1, y_1, α_1) has been memorized, it turns approximately $90°$ and drives away to a specified distance (4). Then it turns back (5) and reapproaches the light-source (6). After reaching the optimal distance again (7), it is possible to calculate the absolute position of the light-source (x_{Lq}, y_{Lq}) with the information from the second position (x_2, y_2, α_2) (see left picture). Finally, the Khepera leaves the

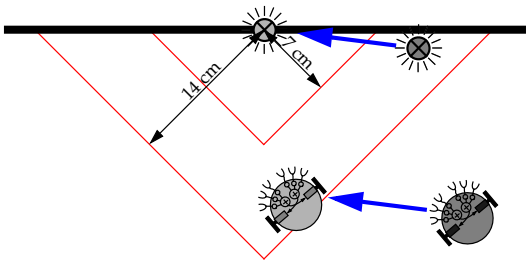


Figure 30. Proportional view of the Khepera and the two circles (in the Manhattan distance).

area of the light-source (8, $\Theta_3 = 490$). The implemented neural network used to approach the light-source, is shown on the right of Fig. 29. The geometrical calculations are carried out according the Eqn. 24 to 26. Note that the drive-to-light-source schema is implemented in analogy to Braitenberg's Vehicle IIIa, which differs from the one implementing the obstacle avoidance behaviour in the excitatory nature of its sensori-motor coupling. Moreover, the triangulation process is independent of the light-source's intensity, as a higher basic radiation will only result in the robot taking the two angles from a greater distance.

$$l_{Lq} = \frac{(x_2 - x_1)\sin(\alpha_1) + (y_1 - y_2)\cos(\alpha_1)}{\sin(\alpha_2 - \alpha_1)} \tag{24}$$

$$x_{Lq} = x_2 + l_{Lq}\cos(\alpha_2) \tag{25}$$

$$y_{Lq} = y_2 + l_{Lq}\sin(\alpha_2) \tag{26}$$

In order to recognize already encountered light-sources, two areas specific to each light-source have been defined (Fig. 30) and are relevant for changing the state of a previously discovered light-source. The outside margin marks the tolerance area in which a light-source is recognized to be identical to the previously recorded one. In this case, the difference between the stored position of the light-source and the current one will be used to adjust the position (x, y) of the Khepera. Moreover, if the robot enters the inner area without detecting a light-source, the one previously registered at this location will be removed from the list of operating light-sources. Eventually, an error propagation calculation (Eqn. 27, 28) was carried out to evaluate the effectiveness of the applied method ($F_{\alpha_{Lq}} = 1,52°$, and number of program cycles between registering the two angles/positions = about 30, e.g. the odometrical error approximately amounts to $F_{x_R} = F_{y_R} = 1.7\,\mathrm{mm}, F_{\alpha_R} = 5.4°$).

$$F_{k_{Lq}} = \left|\frac{dk_{Lq}}{dx_2}\right| F_{x_R} + \left|\frac{dk_{Lq}}{dy_2}\right| F_{y_R} + \left|\frac{dk_{Lq}}{d\alpha_2}\right| \left(F_{\alpha_R} + F_{\alpha_{Lq}}\right) + \left|\frac{dk_{Lq}}{d\alpha_1}\right| F_{\alpha_{Lq}} \tag{27}$$

$$F = \sqrt{\sum_{k=x,y} {F_{k_{Lq}}}^2} \tag{28}$$

Fig. 31 shows the total error versus the angular difference depending on the absolute distance. At position one (x_1, y_1), the light-source is assumed to be detected under an angle of $\alpha_1 = 90°$ (at $\alpha_2 = \pm 90°$, e.g. $|\alpha_1| - |\alpha_2| = 0°$, the algorithm diverges). Fig. 32 shows a trajectory obtained from simulations, which underlines the effectiveness of the applied method, since all errors concerning the light-source locations arise through the neural network used to calibrate the light-sensors and not from odometrical errors (the position calculation in the simulator [26]

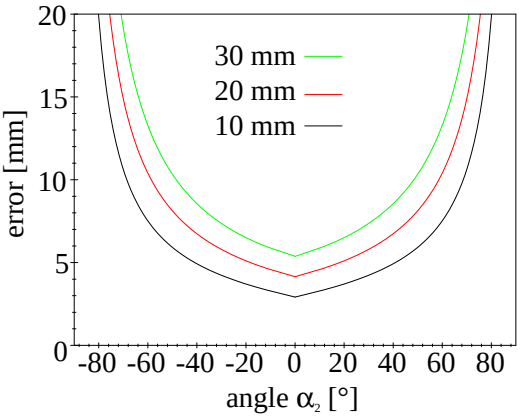


Figure 31. The three curves display the error (Eqn. 28) which occurs at three different distances to the light-source (10 mm, 20 mm, 30 mm).

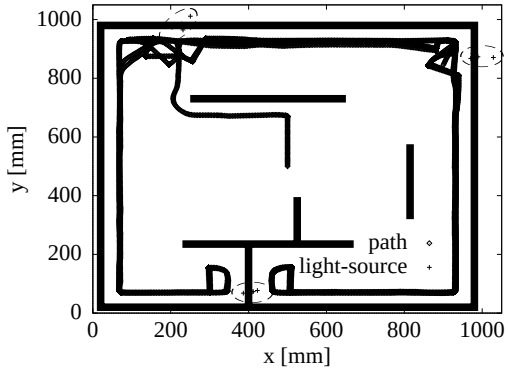


Figure 32. Path of the simulated Khepera which always recognized the three different light-sources correctly.

is assumed to be perfect). Finally, the results from two real experiments (Fig. 33) illustrate the effects of the angular error in the internal position determination, which cannot be corrected by the proposed process. In the case without position correction (Fig. 33 left) the robot was able to recognize the light-source 5 times after the first detection. Using position correction (Fig. 33 right) the experiment was terminated after 24 recognitions. This experiment shows that the position correction using detected light-sources as local landmarks works well, but that still a problem with the angular error remains. This prompts one to investigate further possibilities to obtain a truly 'global' positioning system capable of correcting this kind of error. The next subsection shall discuss this in relation to the application of the distance-calibrated linear camera.

6.2 A Global Positioning System

The methodology described in the following may be divided into two phases: an initialization and operation.
During the first, the locations of the three cylindrical landmarks are determined. Due to the fact that the robot is not able to distinguish between these landmarks by their appearance, they have to be positioned so to allow an unique identification. The solid line (Fig. 34) marks the

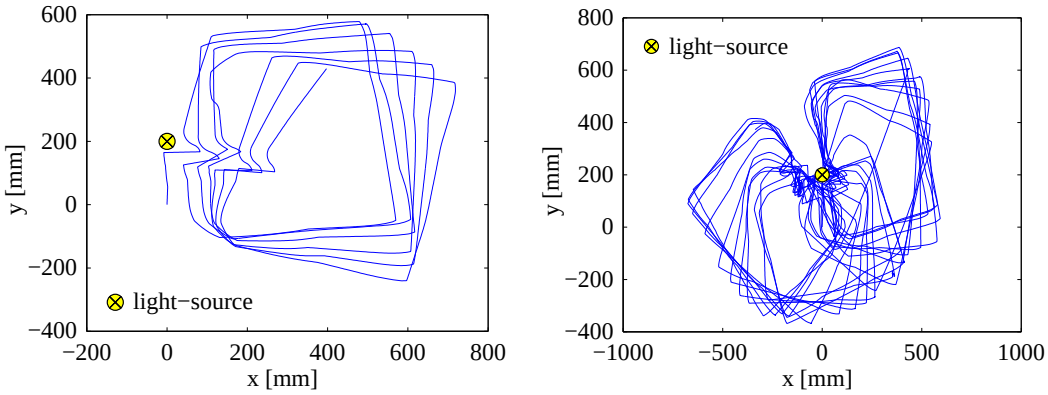


Figure 33. Path of the real Khepera in two experiments for the position adjustment with one singular light source in an area of $75 \times 60\,\mathrm{cm}$.

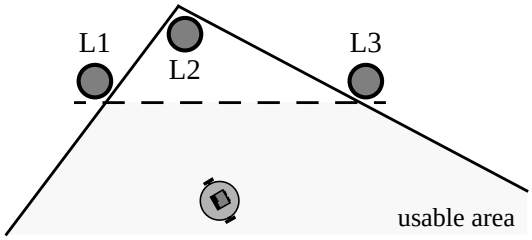


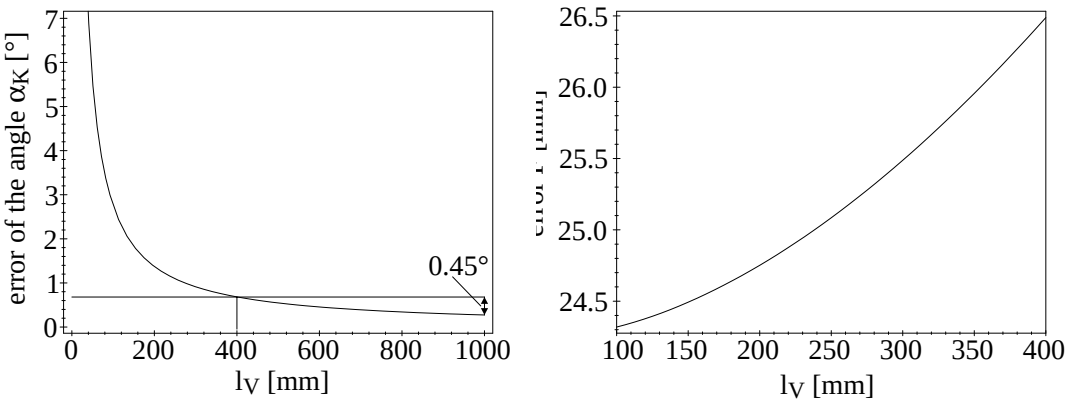
Figure 34. Margins of the area in which the Khepera is able to recognize the landmarks (L1-L3) correctly.

area in which it is possible to discriminate the landmarks. Preventing a cyclic permutation of the landmarks, the angle between the outer ones has to be less then $180°$. Therefore, the Khepera has to keep below the broken line. Moreover, the angular differences under which these extended objects appear is to be determined. This is done by applying the odometrical path integration (α_R) and extracting the angular mid-point position of the perceived landmark in the camera image (Fig. 26). α_K is to be corrected (Eqn. 29) due to the fact that the camera is not exactly located at the robot's geometrical centre. Taking the resolution of the linear camera into account, for distances of more than 400 mm the error can only be estimated (Fig. 35(a)). Therefore a discrepancy of $0.45°$ arises from a distance of 1 m. The landmark's coordinates are finally obtained by means of Eqn. 30 and 31. Eventually a rotation of the coordinate system is performed in order to assure that the robot's relative position with respect to the landmarks is uniquely identifiable, regardless of its initial position and orientation. Furthermore, an error propagation calculation (Eqn. 32) shows that the resulting position error increases with increasing distance (Fig. 35(b)) and is periodically dependent on α_K and α_R (Fig. 36), whereby the elementary uncertainties are evaluated to $F_{l_V} = 24\,\mathrm{mm}$, $F_{\alpha_K} = 36°/64\,\mathrm{pixel}$ and $F_{\alpha_R} \approx 1°$.

$$\alpha_K = \arctan\left(\frac{l_V \sin\left(\alpha'_K\right) + 5\,\mathrm{mm}}{l_V \cos\left(\alpha'_K\right)}\right) \tag{29}$$

$$x_{Li} = x_R + l_V \cos\left(\alpha_R + \alpha_K\right) \tag{30}$$

$$y_{Li} = y_R + l_V \sin\left(\alpha_R + \alpha_K\right) \tag{31}$$



(a) (b)

Figure 35. The corrected error vs. distance from the landmark (a) and the Graph of the error (F) (Eqn. 32) which occurs during the calculation of the landmark's position vs. the distance (l) to the landmark $(\alpha_K = 0°, \alpha_R = 0°)$ (b).

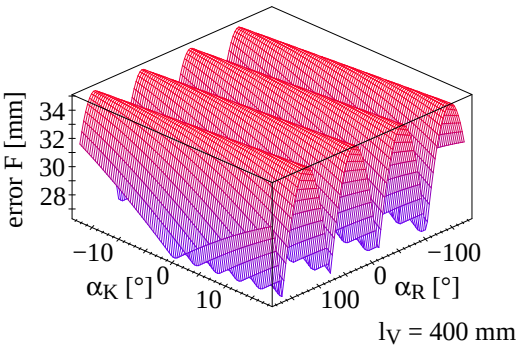


Figure 36. Graph of the error (F) (Eqn. 32) which occurs during the calculation of the landmark's position in dependence of the position of the landmark in the picture (α_K) and the angular orientation of the robot (α_R).

$$F = \sqrt{\sum_{k=x,y} \left(\left| \frac{dk_{Li}}{dl_V} \right| F_{l_V} + \left| \frac{dk_{Li}}{d\alpha_K} \right| F_{\alpha_K} + \left| \frac{dk_{Li}}{d\alpha_R} \right| F_{\alpha_R} \right)^2} \tag{32}$$

During the operation phase, the position/orientation is updated in an event-driven way, whereby an event may, for example, be represented by the encounter of a light-source or of a dead-end structure. Fig. 37 shows the geometrical situation wherein the two difference-angles δ_1 and δ_2 are measured, permitting to calculate x_R, y_R and α_R by means of Eqns. 33 to 44. Note that for $\delta_1 + \delta_2 + \gamma = 180°$, a position calculation is not possible, since Eqn. 38 no longer supplies any information (Fig. 38). The corresponding error propagation calculation (Eqn. 45, $F_\delta = 2\,\text{pixel} \cdot 18°/64\,\text{pixel}$) shows that the applied method provides a good position correction for mid-range distances. In the given example (Fig. 39) the landmarks are placed at the following positions: $L1(0, 1000)$, $L2(200, 1100)$ and $L3(500, 1000)$. The asymmetry in the error is due to the asymmetry in of the landmarks' positions. In order to fulfil the given constraints (Fig. 34), the marked areas are impossible for the robot to reach. Moreover, due to

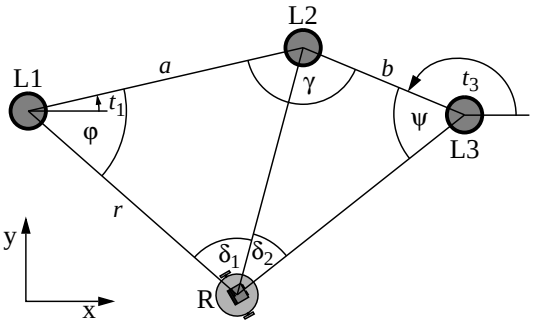


Figure 37. Schematic for the calculation of the robot's position (R) by means of the three landmarks (L1 - L3).

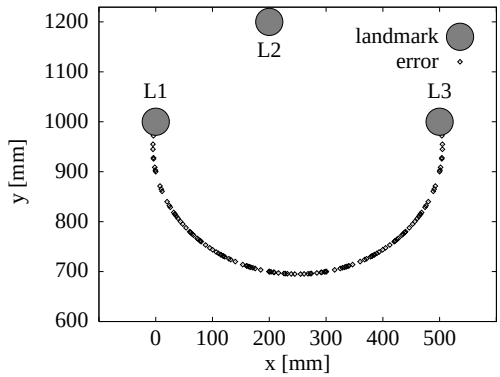


Figure 38. The diagram shows the positions at which a calculation of the position is impossible assuming given locations of the landmarks.

the imprecise mapping of the landmarks' locations during the initialization phase, systematic distortions of the perceived robot-positions arise. Fig. 40 shows the deferral between the real and the internal position of the robot. Exemplarily the middle landmark was deferred 10 mm in y-direction and the right one 25 mm in x-direction. The result shows that the position calculation is very sensitive to such deferrals which may arise throughout the initialization process.

$$t_1 = \arctan\left(\frac{y_{L2} - y_{L1}}{x_{L2} - x_{L1}}\right) \tag{33}$$

$$t_3 = \pi - \arctan\left(\frac{y_{L3} - y_{L2}}{x_{L3} - x_{L2}}\right) \tag{34}$$

$$\gamma = |t_1 - t_3| \tag{35}$$

$$a = \sqrt{(x_{L2} - x_{L1})^2 + (y_{L2} - y_{L1})^2} \tag{36}$$

$$b = \sqrt{(x_{L3} - x_{L2})^2 + (y_{L3} - y_{L2})^2} \tag{37}$$

$$\varphi + \psi = 360^\circ - \gamma - \delta_1 - \delta_2 \tag{38}$$

$$\frac{a \sin(\varphi)}{\sin(\delta_1)} = \frac{b \sin(\psi)}{\sin(\delta_2)} \tag{39}$$

$$\varphi = \arctan\left(\frac{-b \sin(\delta_1) \sin(\gamma + \delta_1 + \delta 2)}{a \sin(\delta_2) + b \sin(\delta_1) \cos(\gamma + \delta_1 + \delta_2)}\right) \tag{40}$$

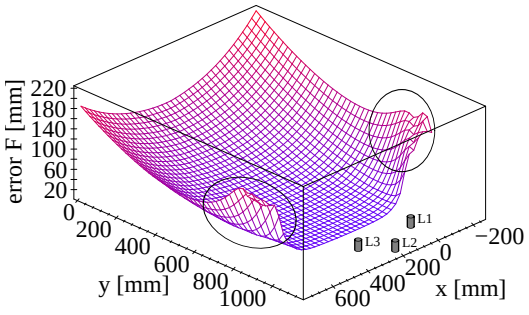


Figure 39. Error (Eqn. 45) due to inaccuracies during the measurement of the angles in dependence of the robot's position.

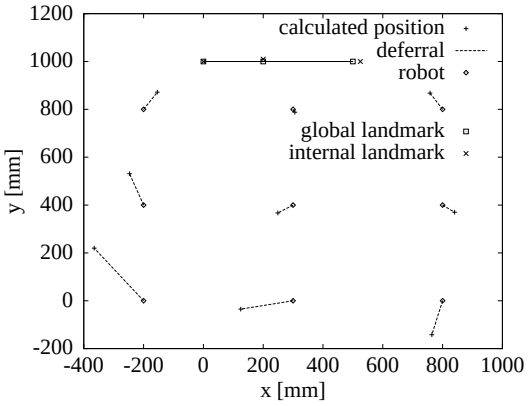


Figure 40. The error which occurs during recording the landmarks' positions (solid line) prevents an exact calculation of the robot's position.

$$r = \frac{a}{\sin(\delta_1)} \sin(\pi - \varphi - \delta_1) \tag{41}$$

$$x_R = x_{L1} + r\cos(t_1 - \varphi) \tag{42}$$

$$y_R = y_{L1} + r\sin(t_1 - \varphi) \tag{43}$$

$$\alpha_R = \arctan\left(\frac{y_{L3} - y_R}{x_{L3} - x_R}\right) - \alpha_K \tag{44}$$

$$F = \sqrt{\left(\left|\frac{dx_R}{d\delta_1}\right| + \left|\frac{dx_R}{d\delta_2}\right|\right)^2 + \left(\left|\frac{dy_R}{d\delta_1}\right| + \left|\frac{dy_R}{d\delta_2}\right|\right)^2} F_\delta \tag{45}$$

7 Discussion of the Foraging Animate Problem

7.1 The Foraging Animate Problem Specification

The Foraging Animate Problem considerably extends the so-called Dynamical Nightwatch's Problem [21], which by itself is already a maximal one. Consequently, this extension is possible only through the additional use of two hardware modules, i.e. the gripper and the vision turret.

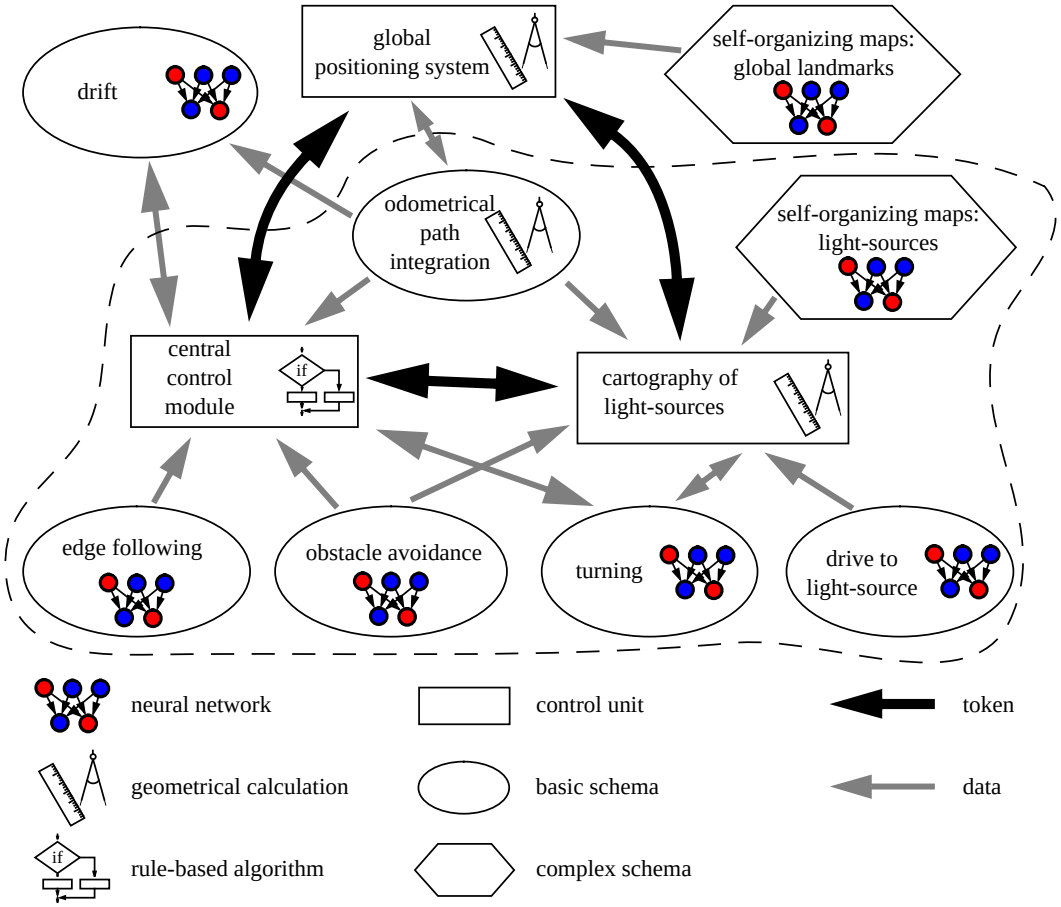


Figure 41. Representation of the interaction schema that realizes a solution to the Foraging Animate Problem.

It may be specified in the following way:
During an initialization phase, the robot registers the positions of the three, tower-like global landmarks (‘skyscrapers’) utilizing the distance information extracted from the vision data. The next step consists of searching for the location of its home-base. This home-base is located at a dead end and labelled by a visual mark. Consequently, it may be recognized by the simultaneous registration of both characteristics (dead-end: infrared sensors, visual mark: linear camera). Subsequently, the robot constantly explores the unknown and potentially changing environment, while looking for light-sources to map (applying the angular information, extracted from the light-sensor data) and registering them. Every time the position of a light-source is registered or deleted, the robot communicates its findings to the human observer by means of the two LEDs. Optionally, a local position correction may be carried out using the difference-vector between the perceived and the memorized position of the light-source in question. The next task consists of transporting an object associated with each switched-on light-source to its home-base by means of the point-to-point navigation algorithm. Having reached the home-base, the object is deposited. Moreover, the position of the robot is updated by an event-driven mechanism, e.g. each time the home-base is reached, using the global landmarks as a reference. After this, a new light-source is sought.

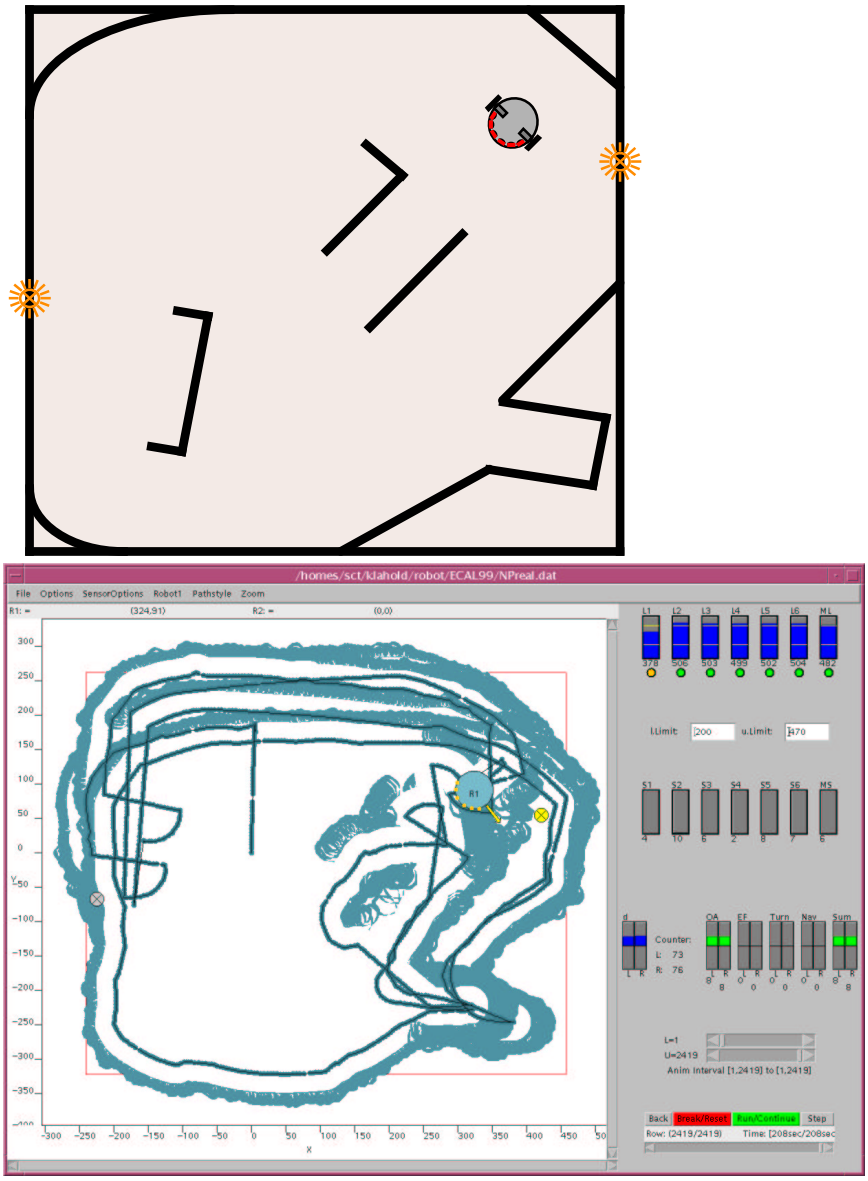


Figure 42. Visualization (bottom) of a complete data set recorded from a real experiment concerning the Dynamical Nightwatch's Problem together with the corresponding environment ($80 \times 80\,\mathrm{cm}$; top). Note the drift of the internal position.

7.2 The Interaction Schema

The interaction schema (Fig. 41) realizes the implementation concept presented in the introduction (Fig. 2). It is an extension of the so-called Dynamical Nightwatch's Problem [21] indicated by the dashed line. Every basic schema operates on a specific part of the available sensor space; only the control units may send commands to the motors. To enhance the modularity of the implemented solution and to better visualize the internal processes, e.g. the token-transfer on the basis of current sensor data and internal states, the central control unit has been trisected. Note that this interaction schema has to be considerably altered—with respect to the Dynamical Nightwatch's Problem—to include the new functionalities, hence increasing the number of possible interactions in a non-linear way.

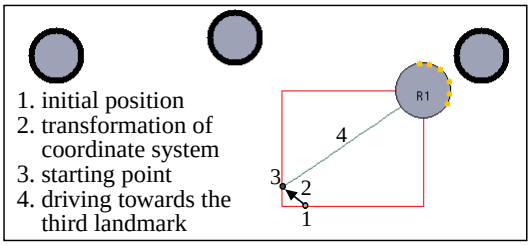


Figure 43. Visualization of the initial landmark mapping in a real experiment.

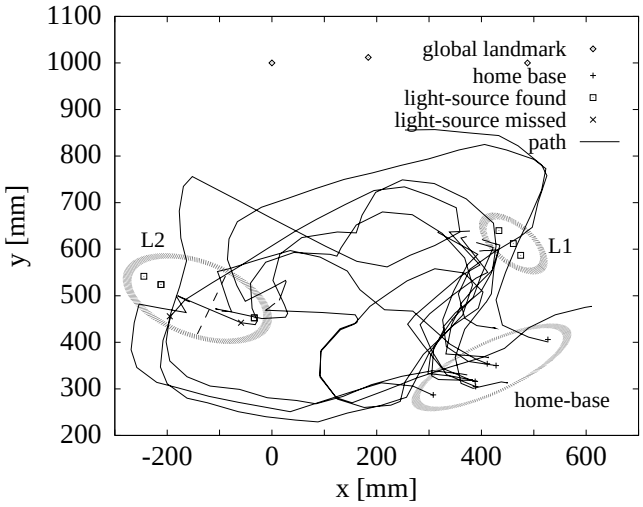


Figure 44. Trajectory of a real Khepera in an unknown environment with two light-sources (L1, L2) and a home-base (the ellipses mark the areas in which the respective object has been recognized).

7.3 Graphical Analysis

The graphical analysis of the obtained behaviour shall be exemplified in two steps. Firstly, a visualization of a real experiment concerning the Dynamical Nightwatch's Problem is represented together with the corresponding environment (Fig. 42). Secondly, the initialization phase concerning the mapping of the three global landmarks is depicted (Fig. 43). In reality, the landmarks have the same x-position and y-distances of 20 and 30 cm respectively.

7.4 Discussion of the Results

Considering the possible errors regarding the positioning system, the test-results (Fig. 44) are satisfactory with respect to the real experiment. The location of the home-base varies due to the errors in the global positioning system causing the registration of light-source L2 at two different points (indicated by the dashed line). Nevertheless, it should be remarked that with the additional hardware modules (gripper and vision turret), the autonomy time, determined by the depletion of the available energy resources, only amounts to approximately a third, whereby—towards the end—some infrared sensor degradations were observed, negatively affecting the obstacle avoidance and edge following behaviours. Moreover, the intensity independence, with respect to light-sources, had to be somehow restricted: gripping the objects associated with

light-sources is to be triggered by an intensity threshold due to the negative influence of nearby light-sources on the infrared-proximity sensors. Sometimes, the robot is not able to grab the object because of the lower lateral angular resolution of the corresponding neural network due to the non-isotropic sensor arrangement. Furthermore, the recognition of the home-base is subjected to two independent sources of noise, i.e. the infrared proximity data and the vision turret data. In an unfavourable situation, this may lead to a non-recognition of the home-base. Finally, there is a possible source of conflict between the two position correction methods; the correction at the light-source location may improve the cartesian components, however, the angular error is not corrected, which enhances the angular maladjustment due to the global positioning system. The latter—on its own—is (cf. section 6) only suitable for mid-range distances.

8 Conclusion and Outlook

In this paper, the implementation of the solution to a maximal robotics problem, i.e. the Foraging Animate, has been presented. Given a small and restricted system like the mini-robot Khepera, a new design methodology as well as the corresponding implementation concept had to be developed in order to ensure a robust, real-time capable and resource-efficient solution. The exploration and navigation algorithms developed, inspired by the work of Braitenberg [1], are—although dependent on internal states—basically reflexive in nature and—despite of the fact that they have been optimized for the Khepera mini-robot—in principle scalable to larger systems, which are potentially more interesting for industrial applications. Moreover, the application of a variant of Kohonen's algorithm for adaptive sensor calibration represents a method to both extract symbolic information from sub-symbolic sensor data and to compensate fabrication tolerances of sensor components. The utilization of the geometrical informations obtained, greatly increased the positioning system accuracy based on odometrical path integration. Finally, a particular focus of this paper has been on the analysis of the sensorial system, the methods applied and the behaviour obtained both by classical error propagation techniques and graphical visualization. These results then allowed the optimization of the corresponding system parameters as well as a better evaluation of the system's performance with respect to the underlying constraints.

After having demonstrated the solution to a single robot's maximal problem, the next logical step consists of investigating maximal cooperative behaviour arising, for example, in the context of cooperative clustering strategies. To successfully handle this kind of problem, the development of a powerful communication module for Khepera mini-robots like the one described in Rueping et. al. [34, 33] is required. Moreover, it is envisaged to map parts, e.g. the exploration algorithms or the Kohonen feature maps used for sensor calibration, of the above described software solution to the FPGA module presented in [38]. Eventually, despite our efforts to fuse three sensorial sources to upgrade the positioning system, the inaccuracy of the positioning process still presents the major short-coming of the present implementation. Therefore, we are currently investigating two different approaches to overcome this deficiency. On the one hand, a matrix of fixed infrared-beacons in conjunction with self-organized data processing is being studied [38], and on the other hand, we are currently examining the potentials of discrete ultrasonic far-range sensors for mini-robots [15].

Acknowledgements. We are very grateful to Prof. G. Domik whose lecture on visualization motivated and supported us to develop a visualization tool, and Ph.D. J. Sitte for intense discussions. In addition, the first author was supported by the DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", GRK 124/2-96.

References

[1] **V. Braitenberg**. *Vehicles: Experiments in Synthetic Psychology*. MIT Press/Bradford Books, 1984.

[2] **R.A. Brooks**. A Robust Layered Control System for a Mobile Robot. *IEEE Transactions on Robotics and Automation*, 2:14–23, 1986.

[3] **R.A. Brooks**. Intelligence without Representation. *Artificial Intelligence*, 47:139–159, 1987.

[4] **R.A. Brooks**. Intelligence without Reason. In *Proceedings of 12th International Joint Conference on Artificial Intelligence*, pp. 569–595, 1991.

[5] **C.C. de Sá**, **G. Bittencourt**, and **N. Omar**. An Autonomous Agent Architecture and the Locomotion Problem. In *Proceedings of the 14th Brazilian Symposium on Artificial Intelligence*, pp. 11–20, 1998.

[6] **D. Floreano** and **F. Mondada**. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11:1461–1478, 1998.

[7] **D. Floreano** and **S. Nolfi**. Adaptive Behaviour in Competing Co-Evolving Species. In *Proceedings of the 4th European Conference on Artificial Life*, pp. 378–387. MIT Press, July 1997.

[8] **D. Floreano** and **J. Urzelai**. Evolution of Neural Controllers with Adaptive Synapses and Compact Genetic Encoding. In *Proceedings of the 5th European Conference on Artificial Life*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pp. 183–194. Springer, 1999.

[9] **M.O. Franz**, **B. Schölkopf**, and **H.H. Bülthoff**. Homing by Parameterized Scene Matching. In *Proceedings of the 4th European Conference on Artificial Life*, pp. 236–245. MIT Press, July 1997.

[10] **G. Hartmann** and **Wehner R.** The ant's path integration system: a neural architecture. *Biological Cybernetics*, 73:483–497, 1995.

[11] **P. Hoppen**. *Autonome Mobile Roboter. Echtzeitnavigation in bekannter und unbekannter Umgebung*. Spektrum, 1992.

[12] **C. Icking** and **R. Klein**. Competitive Strategies for Autonomous Systems. Technical Report 175, Department of Computer Science, FernUniversitt Hagen, 1995.

[13] **N. Jakobi**. Running Across the Reality Gap: Octopod Locomotion Evolved in a Minimal Simulation. In *Proceedings of the 1st European Workshop: EvoRobot98*, pp. 39–57, 1998.

[14] **K-Team S.A.** Products: Mobile Robots. `http://www.k-team.com/products.html`, 1998.

[15] **J. Klahold**, **A. Löffler**, and **U. Rückert**. Discrete Ultrasonic Sensors for Mobile Autonomous Systems. In *Proceedings of the 1st International Khepera Workshop*, volume 64 of *HNI-Verlagsschriftenreihe*, pp. 171–180, 1999.

[16] **T. Kohonen**. *Self-Organization and Associative Memory*. Springer, 2nd edition, 1988.

[17] **A. Kurz**. Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26:233–242, 1996.

[18] **D. Lambrinos**. Navigating with an Adaptive Light Compass. In *Proceedings of the 3rd European Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pp. 602–613. Springer, 1995.

[19] **A. Löffler**, **J. Klahold**, **M. Hußmann**, and **U. Rückert**. A Visualization Tool for the Mini-Robot Khepera: Behavior Analysis and Optimization. In *Proceedings of the 5th International Conference on Artificial Life*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pp. 329–333. Springer, 1999.

[20] **A. Löffler**, **J. Klahold**, **M. Hußmann**, and **U. Rückert**. Demonstration of a Visualization Tool for the Mini-Robot Khepera. WWW-publication of the 5th International Conference on Artificial Life, 1999. `http://diwww.epfl.ch/lami/ecal99/demos/demo4.html`.

[21] **A. Löffler**, **J. Klahold**, and **U. Rückert**. The Dynamical Nightwatch's Problem Solved by the Autonomous Micro-Robot Khepera. In *Proceedings of the 3rd European Conference on Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pp. 303–313. Springer, 1998. This work was awarded with the first prize at international Khepera contest, (results: `http://diwww.epfl.ch/lami/team/michel/khep-sim/contest-results.html`).

[22] **A. Löffler**, **J. Klahold**, and **U. Rückert**. Artificial Neural Networks for Autonomous Robot Control: Reflective Navigation and Adaptive Sensor Calibration. In *Proceedings of the 6th International Conference on Neural Information Processing*, pp. 667–672, 1999.

[23] **A. Löffler**, **F. Mondada**, and **U. Rückert**, editors. *Experiments with the Mini-Robot Khepera, Proceedings of the 1st International Khepera Workshop*, volume 64 of *HNI-Verlagsschriftenreihe*, December 1999.

[24] **P. Maes**. Modeling Adaptive Autonomous Agents. *Artificial Life*, 1(1&2):135–162, 1994.

[25] **K. Malmstrom**, **L. Munday**, and **J. Sitte**. A Simple Robust Robotic Vision System using Kohonen Feature Mapping. In *Proceedings of the 2nd IEEE Australia and New Zealand Conference on Intelligent Information Systems*, pp. 135–139, 1994.

[26] **O. Michel**. Khepera Simulator 2.0. `http://diwww.epfl.ch/lami/team/michel/khep-sim/`, 1998.

[27] **O. Miglino**, **H.H. Lund**, and **S. Nolfi**. Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 4(2):417–434, 1995.

[28] **F Mondada**, **E. Franzi**, and **A. Guignard**. The Development of Khepera. In *Proceedings of the 1st International Khepera Workshop*, volume 64 of *HNI-Verlagsschriftenreihe*, pp. 7–13, 1999.

[29] **F Mondada**, **E. Franzi**, and **P. Ienne**. Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms. In *Proceedings of the 3rd International Symposium on Experimental Robotics*, pp. 501–513. Springer, 1994.

[30] **S. Oore**, **G.E. Hinton**, and **G. Dudek**. A Mobile Robot That Learns Its Place. *Neural Computation*, 9:683–699, 1997.

[31] **A.G. Pipe**, **B. Carse**, and **T.C. Fograty**. Learning Subjective "Cognitive Maps" in the Presence of Sensory-Motor Errors. In *Proceedings of the 3rd European Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pp. 463–476. Springer, 1995.

[32] **E. Prem**. Grounding and the Entailment Structure in Robots and Artificial Life. In *Proceedings of the 3rd European Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pp. 39–51. Springer, 1995.

[33] **S. Rüping**, **A. Löffler**, **C. Odenbach**, and **U. Rückert**. Khepera Module for Wireless Infrared CAN Communication. In *Proceedings of the 1st International Khepera Workshop*, volume 64 of *HNI-Verlagsschriftenreihe*, pp. 181–187, 1999.

[34] **S. Rüping**, **W. Lücking**, and **U. Rückert**. A Wireless Communication System for Khepera Micro-Robots using CAN. In *Proceedings of the European Telemetry Conference*, pp. 100–108, 1998.

[35] **C. Schreier**, **R. Pfeifer**, and **Kunyioshi Y.** Embedded neural networks: exploiting constraints. *Neural Networks*, 11:1551–1569, 1998.

[36] **J. Tani** and **Fukumura N.** Self-organizing Internal Representation in Learning of Navigation: A Physical Experiment by the Mobile Robot YAMABICO. *Neural Networks*, 10:153–159, 1997.

[37] **P.F.M.J. Verschure** and **T. Voegtlin**. A bottom up approach towards the acquisition and expression of sequential representations applied to a behaving real-world device: Distributed Adaptive Control III. *Neural Networks*, 11:1531–1549, 1998.

[38] **U. Witkowski**, **A. Heittmann**, and **U. Rückert**. Implementation of Application Specific Neural Hardware on the Mini Robot Khepera. In *Proceedings of the 1st International Khepera Workshop*, volume 64 of *HNI-Verlagsschriftenreihe*, pp. 189–196, 1999.

[39] **T. Ziemke**. Towards Adaptive Perception in Autonomous Robots Using Second-Order Recurrent Networks. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, pp. 89–98, 1996.

A Nomenclature

Odometrical Path Integration

α_R, α_{R_0}	Orientation of the robot with respect to the initial x-axis at t and t_0 with $t = t_0 + \Delta t, \Delta t > 0$
$(x_R, y_R), (x_{R_0}, y_{R_0})$	Cartesian coordinates at t and t_0
n_r, n_L	Incremental encoder values of the right and left motor: number of counted wheel-step pulses per Δt
Δl	Advancement per puls of the incremental encoders
d	Diameter of the robot
r	Rotation radius with respect to one integration step

$\Delta x_R, \Delta y_R, \Delta \alpha_R$	By n_R, n_L induced alterations of x_R, y_R and α_R
F_{n_R}, F_{n_L}	Estimated error of the incremental encoder values during Δt
F, F_x, F_y, F_α	Position- (total-, resp. x-, y- and α-error) and angular error with respect to one integration step

Adaptive Sensor Calibration

f	Function to be approximated, which possesses an intrinsic monotony to be preserved by the training algorithm
$\mathbb{R}$	Set of real numbers
$\vec{s} = (s_0 \dots s_n)$	n-dimensional vector (infrared sensor values)
o	Scalar output value
$\vec{V}_{ex}$	(n+1)-dimensional example vector
i, bm	Index of a neuron
$\vec{w}_i$	Weights of the i-th neuron
φ_i	Associated output value of the i-th neuron
$\vec{V}_i$	Vector representing the i-th neuron
$\vec{V}_{bm} = (\vec{s}_{bm}, o)$	Best match vector (with respect to the 1-norm)
$\lambda\,(t, \lvert i - bm \rvert)$	Learning rate of the neuron
$\vec{V}_{re}$	n-dimensional recall vector

Cartography of Light-Sources

(x_1, y_1, α_1)	Position of robot and direction to light-source at t_1
(x_1, y_1, α_2)	Position of robot and direction to light-source at $t_2 > t_1$
l_{Lq}	Distance to light-source at t_2
(x_{Lq}, y_{Lq})	Position of light-source
$F_{k_{Lq}}, F$	Error of calculated position of light-source in k-direction $(k = x, y, \alpha)$ resp. total error
Θ_i	Intensity thresholds $(i = 1, 2, 3)$

Global Positioning System

B_i	Grey-scale level of the i-th pixel of the vision turret
G_i	Difference in grey-scale level between the i-th and the (i+1)-th pixel
α'_K, α_K	Apparent and corrected angle under which the robot perceives a global landmark
l_V, F_{l_V}	Distance vision turret - mid-point of global landmark, resp. the error
(x_{Li}, y_{Li})	Position of the i-th global landmark $(i = 1, 2, 3)$
$t_1, t_3, \gamma, \varphi, \psi$	Auxiliary angles used in the position calculation based on global landmarks
a, b	Distances of the three global landmarks
r	Distance between robot and first global landmark
$\delta_1, \delta_2, F_\delta$	Perceived angular difference with respect to the positions of the three global landmarks, resp. the error