Controlling a Robot With Image-based Homing

Thomas Röfer

Center for Cognitive Sciences
University of Bremen
P.O. Box 330 440
D-28334 Bremen

E-mail: roefer@informatik.uni-bremen.de
www: www.uni-bremen.de/~roefer/main.html
Phone: +49 421 218 3495
Fax: +49 421 218 3054

Abstract. This paper presents a biologically inspired method for the navigation of autonomous mobile systems. The method calculates the way from a current position to a target position using one-dimensional 360° images, taken at these positions. The correlations between the two images are generated by using a modified version of Kohonen's self-organizing feature map. The direction to the target position is calculated from the weight vectors of the trained network. Experiments with a small robot prove the method to be stable and exact.

1 Introduction

The navigation of autonomous mobile robots is still an unsolved problem. Robots operate in a three-dimensional, changing world. In this environment they have to navigate in real time. For this reason there has to be a suitable technique for navigation. Methods that are based on three-dimensional models are very complex: it takes a lot of time to match the sensor data with three-dimensional objects stored in large databases. Therefore it seems to make sense to search for methods that are less complicate. One possibility is to follow biological examples because even very "simple" animals are able to navigate very exactly.

Cartwright and Collett (1983, 1987) analyzed the navigation of honeybees. They supposed that the bees navigate with the help of "snapshots". With images of the environment the bees remember the exact position of the beehive and their food sources. If they approach such a position they use the differences between the current image and the target image to get back to their beehive or food source.

Cartwright and Collett presented an algorithm that works only with very simple, binary images and probably not with real world data. In contrast to this the approach presented in this paper uses real images taken by a one-dimensional 360° sensor to control a mobile robot.

2 The Robot

A small three-wheeled robot was built as a testbed for developing a navigation method that works in the real world. It was built with parts of a toy robot kit (Fischertechnik) and is shown in Figure 1. It is controlled by an external IBM compatible personal computer to which it is connected with a cable.

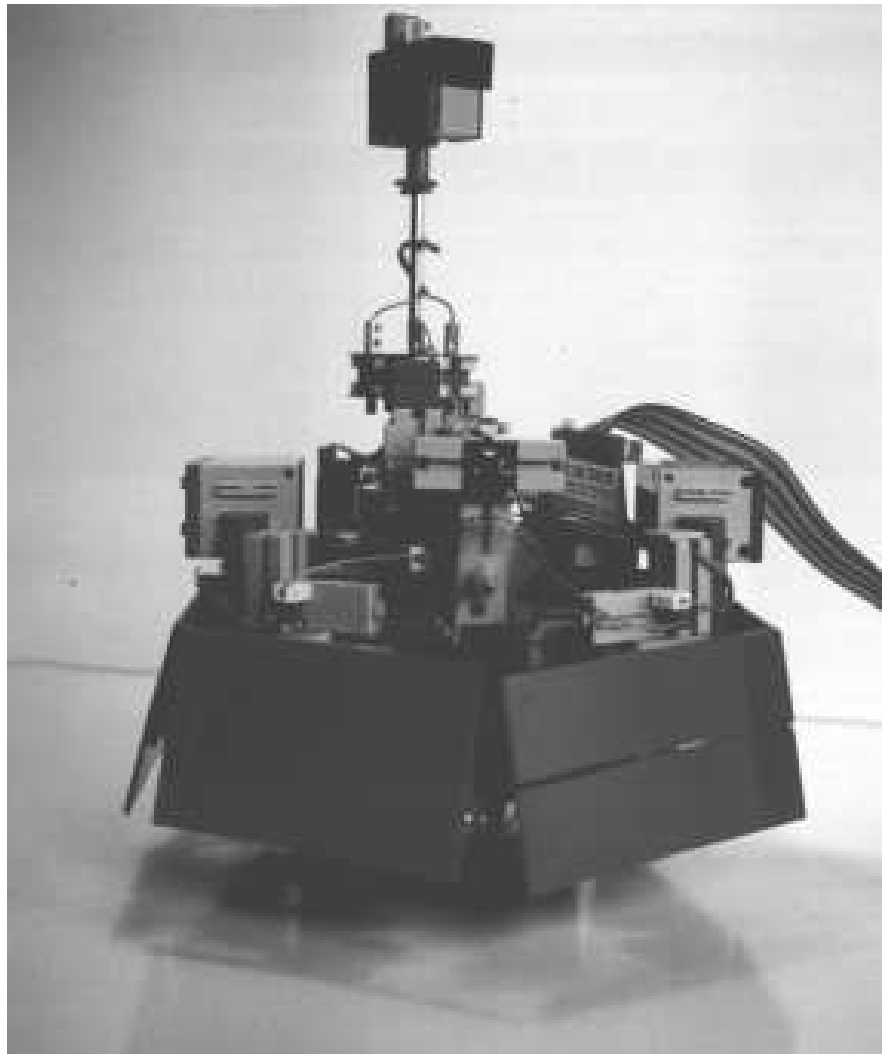

Figure 1: The robot

The chosen construction of the robot leads to three essential characteristics:

- The robot is able to drive in different directions without turning its body. That is achieved by synchronous steering of all three wheels. Therefore the wheels always point in the same direction.
- A rotating photoreceptor is fixed to the top of the robot. It measures 64 values of brightness in one turn. This is comparable to 64 separate sensors that are arranged in a circle. With each turn the sensor takes a 360° image of its environment.
- One surrounding bumper supports simple obstacle avoidance.

3 Matching Images

Examples of the images that are taken by the brightness sensor are shown in Figure 2 and Figure 3. Although both images are taken from positions, that are only 50 cm apart, there are many differences between them: the first maximum in the second image is brighter than in the first one. Also the second maximum in Figure 3 is wider than in Figure 2. A maximum exists in the middle of the second image that can hardly be seen in the first image.

To navigate with the help of these images it is necessary to assign which part of the first image belongs to which part of the second image. Hong et al. (1991) developed an algorithm that searches for characteristic points in one image and tries to find them in the second image. Characteristic points are those points with maximum intensity change. During the searching process, normalized small windows of intensity are compared to each other. Because of the normalization there are many confusions. Hong tried to eliminate these confusions by reducing the search area for each characteristic point.

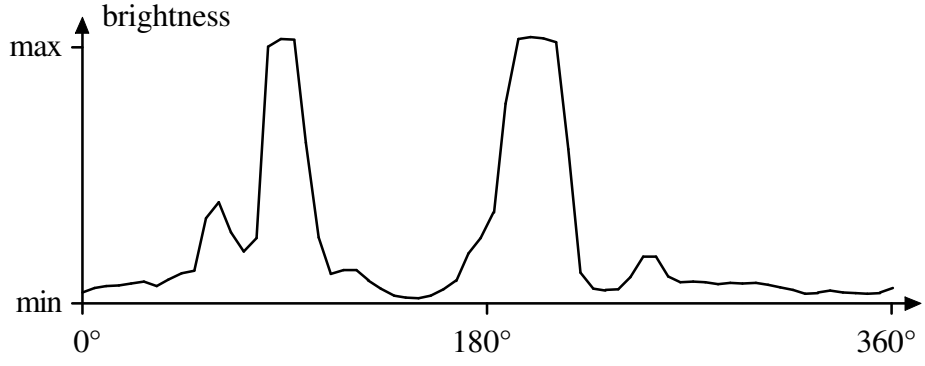

Figure 2: Typical image of a target position

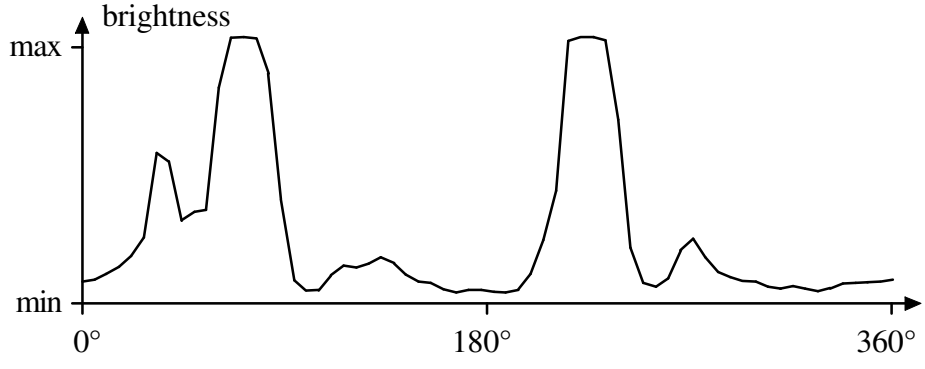

Figure 3: Image of a position 50 cm away from the target

It seems that the intensity in a small window is not sufficient to make correct correlations. For this reason the method presented in this paper has been developed. It uses much more informations to achieve the correlations. The algorithm uses the brightness, the first derivation and the second derivation of the brightness to match the images. It also uses the topology of the pixels. It assumes that neighboring points in one image are

neighbors in the other, too. The algorithm is realized with a modified version of Kohonen's (1982) self-organizing feature map.

4 The Modified Kohonen Algorithm

The goal of the matching step is to find the correspondence between the pixels of the target image I^T and those of the current image I^C. The idea is to initialize a modified Kohonen-network with the target image and to train it with the current image. After a successful training, the weight vectors of the target image have moved to corresponding vectors in the current image. To maintain the identity of the weight vectors, only the dimension, that codes the position, is changed during training. The matching of the images in Figure 2 and Figure 3 is shown in Figure 4.
The difference between the original position and the position in the trained network is the shift of a pixel. So the algorithm calculates a shift for each pixel of the image. It uses a one-dimensional Kohonen-network with four-dimensional weight vectors. Two modifications to Kohonen's algorithm were made:

- Because the images are 360° views, the neighborhood relation and the distance relation of the weight vector's dimension, that codes the position, are those of a ring.
- Because the topology is predefined in the image, the learning rule always preserves it so that the dimension of w_i, that codes the position, is always between those of w_{i-1} and w_{i+1}.

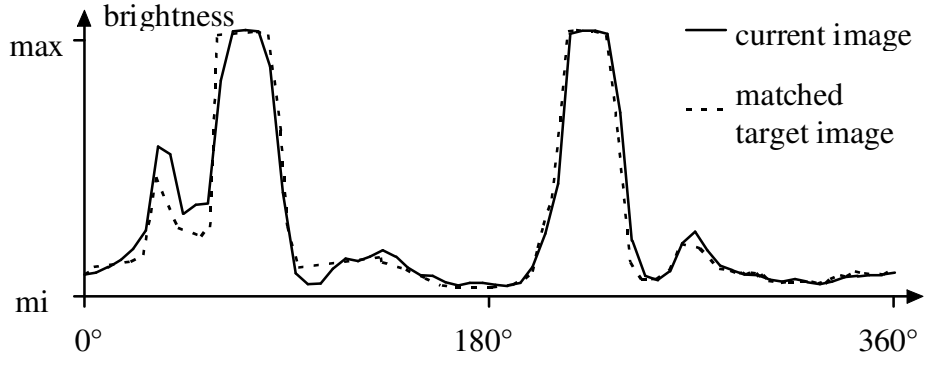

Figure 4: Matching of target image to current image

4.1 Initialization

The network is initialized and trained with vectors $\Psi_{I,i}$. They consist of the following information about the pixel I_i of image I:

- The pixel's index. This is equivalent to the direction, in which this pixel is seen. The index is translated into the range of [0,1[which is called the pixel's position. 0 stands for 0° and 0.5 for 180°.
- The pixel's brightness in the range of [0,1[.
- The first derivation of brightness.

- The second derivation of brightness.

The derivations are calculateted using a filter. p is the number of pixels in the image:

$$\Psi_{\mathrm{I,i}} = \left(\frac{i}{p}, I_{\mathrm{i}}, I'_{\mathrm{i}}, I''_{\mathrm{i}} \right) \tag{1}$$

All the net's weight vectors w_{i} are initialized with the target image I^{T}:

$$\underset{\mathrm{i}}{\forall} w_{\mathrm{i}} = \Psi_{I^{\mathrm{T}},\mathrm{i}} \tag{2}$$

4.2 Training of the Network

The network is trained in $t_{\max}$ steps with randomly chosen pixels of the current image I^{C}.

$$v_{\mathrm{t}} = \Psi_{I^{\mathrm{C}},\mathrm{random}} \tag{3}$$

For each training vector v_{t} the best matching weight vector w_{best} is calculated:

$$\left| v_{\mathrm{t}} - w_{\mathrm{best}} \right| = \min_{\mathrm{i}} \left| v_{\mathrm{t}} - w_{\mathrm{i}} \right| \tag{4}$$

During training the learning parameter σ is decreased from nearly half of the number of pixels to one. σ determines how many neighbors of the best matching weight vector are changed during the learning step.

$$\sigma_{\mathrm{t}} = 1 + \frac{t_{\max} - t}{t_{\max}} \left(\frac{p}{2} - 2 \right) \tag{5}$$

If v_{t} lies in the range $]w_{\mathrm{best}-\sigma}, w_{\mathrm{best}+\sigma}[$, the weight vectors with indices in the range of $[best-\sigma, best+\sigma]$ are trained. Only the weight vector's first dimension ($w_{\mathrm{i},0}$), that codes the position, is changed. All weight vectors are shifted in direction of $v_{\mathrm{t},0} \dot{-} w_{\mathrm{best},0}$. Their position between $w_{\mathrm{best}-\sigma}$ and $w_{\mathrm{best}+\sigma}$ determines how far they are shifted. The learning step never changes the ordering of the positions. Dotted versions of + and - consider the arithmetic of a ring.

$$\begin{aligned}
\underset{\mathrm{i} \in [\mathrm{best}-s_{\mathrm{t}}, \mathrm{best}]}{\forall} w_{\mathrm{i},0}^{\mathrm{new}} &= v_{\mathrm{t},0} \dot{-} \frac{v_{\mathrm{t},0} \dot{-} w_{\mathrm{best}-\mathrm{s},0}}{w_{\mathrm{best},0} \dot{-} w_{\mathrm{best}-\mathrm{s},0}} \left(w_{\mathrm{best},0} \dot{-} w_{\mathrm{i},0} \right) \\
\underset{\mathrm{i} \in [\mathrm{best}+1, \mathrm{best}+s_{\mathrm{t}}]}{\forall} w_{\mathrm{i},0}^{\mathrm{new}} &= v_{\mathrm{t},0} \dot{+} \frac{w_{\mathrm{best}+\mathrm{s},0} \dot{-} v_{\mathrm{t},0}}{w_{\mathrm{best}+\mathrm{s},0} \dot{-} w_{\mathrm{best},0}} \left(w_{\mathrm{i},0} \dot{-} w_{\mathrm{best},0} \right)
\end{aligned} \tag{6}$$

5 Computing a Move

Figure 5 shows a typical scene. The robot has taken an image at the target position and stored it. A point in the world has been mapped to pixel $\mathrm{P_i}$ in this image. Later, when it is at another position and tries to return to the target position, the same point is mapped to

pixel P_j. The difference *i-j* between these pixel positions is the shift of the pixel. The goal of the robot is to reduce this shift to zero for all pixels in the image. If there is no shift anymore, the robot has reached the target position.

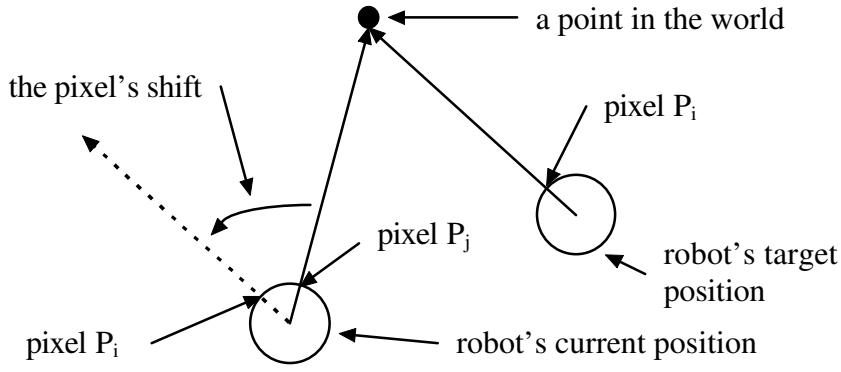

Figure 5: Pixel positions in the robot's current and target positions

A pixel's shift can be easily calculated from the trained Kohonen-network: Equation (1) implies that its original position in the target image can be calculated from its index. Its current position is coded in the first component $w_{i,0}$ of its weight vector. Therefore the pixel's shift s_i can be calculated as follows:

$$s_i = \frac{i}{p} \div w_{i,0} \tag{7}$$

In contrast to Hong's algorithm, the shift can be calculated with a precision that is better than the distance between neighboring pixels.

5.1 Determining the Robot's Own Rotation

The robot can calculate the homing direction from the shift of the pixels. But this works only if the current image is not twisted against the target image. Even though the robot is constructed to move in all directions without turning its body, there are inaccuracies, resulting from the ground the robot is moving on. Before the robot can calculate its moving direction it has to estimate the rotation between the current image and the target image. This is shown in Figure 6.
If there is no twist between the two images, all shifts point away from the target direction. Therefore half of the shifts are positive and half are negative. In this case the sum over all shifts is zero. If there is a rotation, the sum is not zero. Then the average of all shifts is the robot's twist:

$$\alpha = \frac{1}{p} \sum_i s_i \tag{8}$$

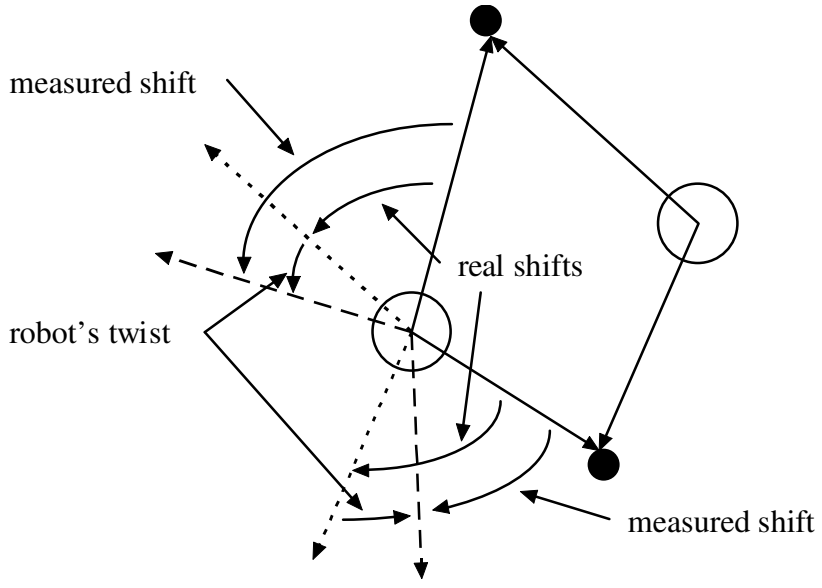

Figure 6: Eliminating the robot's twist.

With this information, the robot can calculate an adjusted shift s_i':

$$s_i' = s_i - \alpha \tag{9}$$

5.2 Calculating the Driving Direction

Figure 7 illustrates the algorithm that calculates a homing vector for the shift of each pixel in the image.

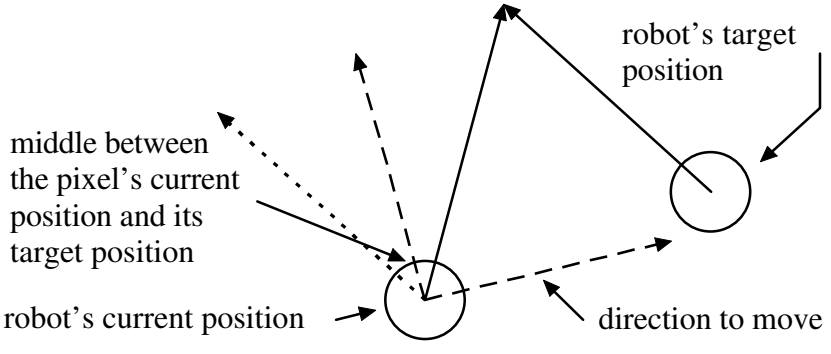

Figure 7: Calculating the homing direction

Like in Hong's paper, it is assumed, that the robot most quickly reduces a pixel's shift, if it moves in a direction, that is perpendicular to the middle between the pixel's position in the current image and in the target image.
A homing vector is formed for each pixel in the image. Because the distance to the target position is unknown, the shift's size s_i' is used as the vector's length. By this larger shifts are emphasized. All these homing vectors are added up. The angle of the resulting vector determines the direction that leads to the target position:

$$\beta = \tan^{-1} \frac{\sum_i s_i' \sin 2\pi \left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4} \right)}{\sum_i s_i' \cos 2\pi \left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4} \right)} \tag{10}$$

5.3 Defining a Stop Criterion

Because the robot cannot calculate the distance to the target position, it has to work incrementally. It computes the homing vector, moves a little bit in this direction, computes a new direction and so on. So there has to be a stop criterion, that tells the robot that it has reached the target location. Besides the robot needs to know how far it has to drive in one step, because it is necessary that it makes smaller steps, if it is near to the target position, so it can reach it precisely. For both the robot uses the length γ of the homing vector, that is described in section 5.2.

$$\gamma = \sqrt{\left(\sum_i s_i' \sin 2\pi \left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4} \right) \right)^2 + \left(\sum_i s_i' \cos 2\pi \left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4} \right) \right)^2} \tag{11}$$

To calculate the distance that the robot covers in one step, γ is multiplied with a constant factor. The robot stops, if γ is less than a predefined value, for example 0.2.

6 Control Architecture

The robot has a hierarchical control architecture. In this hierarchy the sensor data flows bottom-up and the motor command information flows top-down. Modules on each level of the hierarchy can modify the sensor data or replace the motor commands of higher levels. It is even possible to implement new *virtual* sensors that can be used by higher levels.
In the moment there are only six modules in the robot's control architecture that are also shown in Figure 8:

Robot: This module directly controls the robot. It determines the sensor's states and executes the motor commands. It provides two sensorical informations to higher modules: the image detected by the rotating head and the state of the surrounding bumper. The only known motor command has two parameters: distance and direction.

RobotServer, RobotClient: These modules transmit the data over a serial line. **RobotClient** has the same interface as **Robot**.

AvoidObstacles: This module implements a simple obstacle avoidance. If the bumper is active the robot tries to drive in a random direction to bypass the obstacle. While this process motor commands of higher modules are ignored.

Target: This module creates a new virtual sensor for higher modules, the *target image*. In the actual version it stores only one target image if a special button is pressed on the computer. In later implementations it will be able to store and recall sequences of target images. If the distance component of a motor command from a higher module

is less than a certain value, **Target** assumes that the robot has reached the target position and stops it.

Homing: This module performs the homing process. It uses the current image and the target image (generated by the module **Target**) and calculates the homing motor commands.

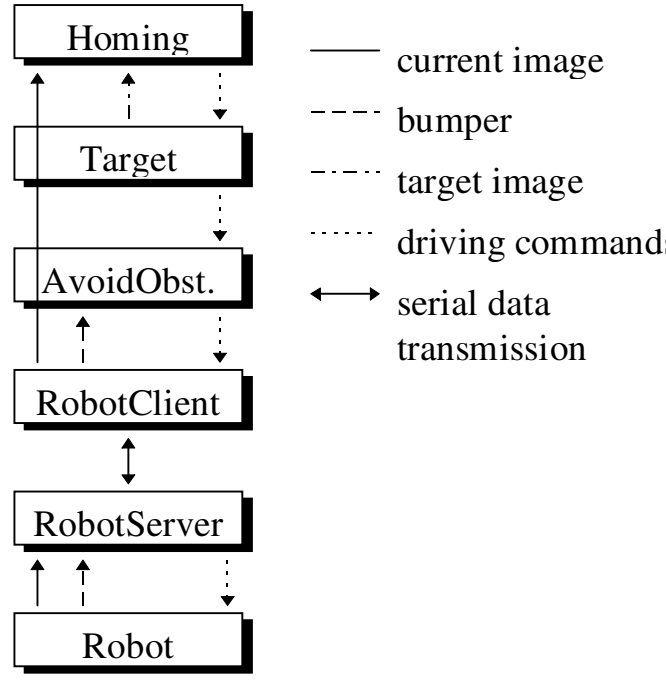

Figure 8: "Subsumption like" control architecture.

7 Experiments

Experiments were made in an environment with three lamps. The robot was put at a random position. From this position it took an image and stored it. Then it was placed to another position, that was up to 1 m apart. After the robot was started, it drove until the stop criterion was reached. The robot always reached the target position with an error less than 3 cm. In the first homing step the network was trained in 1000 learning steps. In all further homing steps only 200 learning steps were needed to update the network.

8 Future Work

There are some points of departure for future work:

8.1 The Integration of Color

To improve the performance of the algorithm color images may be used instead of monochrome images. In first experiments a second photoreceptor was fixed to the rotating head. One receptor measures red light, the other one green light. The network is trained with the brightness, first and second derivations of both sensors and the differ-

ence of their brightnesses. It seems that this eleminates the necessity of viewable lamps in the robot's environment.

8.2 Multiple Target Positions

The robot should use sequences or even networks of neighboring target positions to move over long distances. It may be possible that it picks out its target images by itself while the user teaches trajectories with a joystick.

8.3 The Usage of Distance Information

Many mobile robotic platforms have distance detectors e.g. ultrasonic or infrared sensors. Most often they are used for obstacle avoidance. But if such detectors are fixed regulary around the robot, the measured data can possibly be integrated in the homing method as additional components in the network's weight vectors. This may stabilize homing under changing light conditions. Simulated distance scans are shown in Figure 10 and Figure 11. The room where the distance scans were taken is shown in Figure 9.

9 Conclusion

A biologically inspired method for the navigation of autonomous mobile systems has been presented. The algorithm calculates the way from a current position to a target position using one-dimensional 360° images, taken at these positions. The correlations between the two images are generated by using a modified version of Kohonen's self-organizing feature map. The direction to the target position is calculated from the weight vectors of the trained network. In contrast to other methods it calculates shifts for all pixels of the image. The robot's rotation can be easily eliminated, because the shift information is distributed evenly over the whole image.

Acknowledgements

T. Röfer was supported by the Deutsche Forschungsgesellschaft and the Freie Hansestadt Bremen through the Graduiertenkolleg "Raumorientierung und Handlungsorganisation autonomer Systeme".

References

Cartwright, B. A., Collett, T. S.: Landmark Learning in Bees, Journal of Comparative Physiology 151 (1983) 521-543

Cartwright, B. A., Collett, T. S.: Landmark Maps for Honeybees, Biological Cybernetics 57 (1987) 85-93

Hong, J., Tan, X., Pinette, B., Weiss, R., Riseman, E. M.: Image-based Homing, Proc. of the 1991 IEEE International Conference on Robotics and Automation (1991) 620-625

Kohonen, T.: Self-organized Formation of Topologically Correct Feature Maps, Biological Cybernetics 43 (1982) 59-69

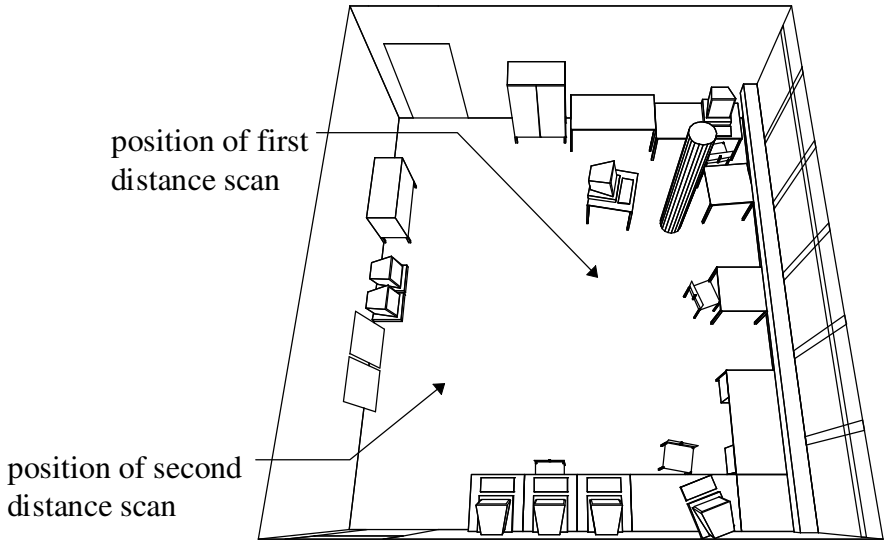

Figure 9: Simulated scene for distance scans.

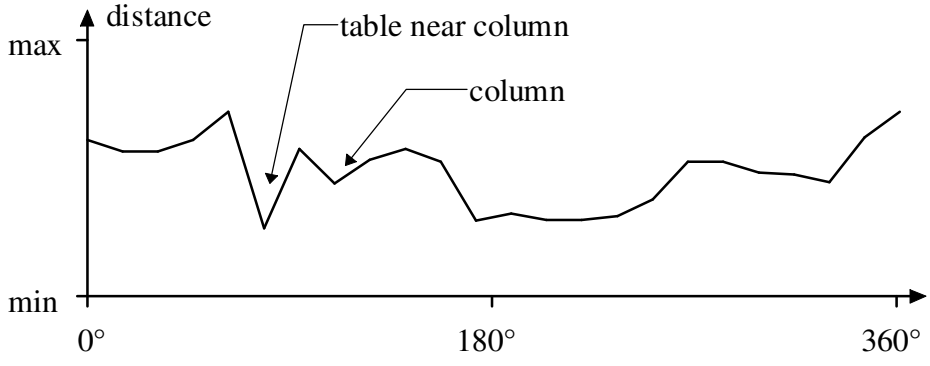

Figure 10: First distance scan.

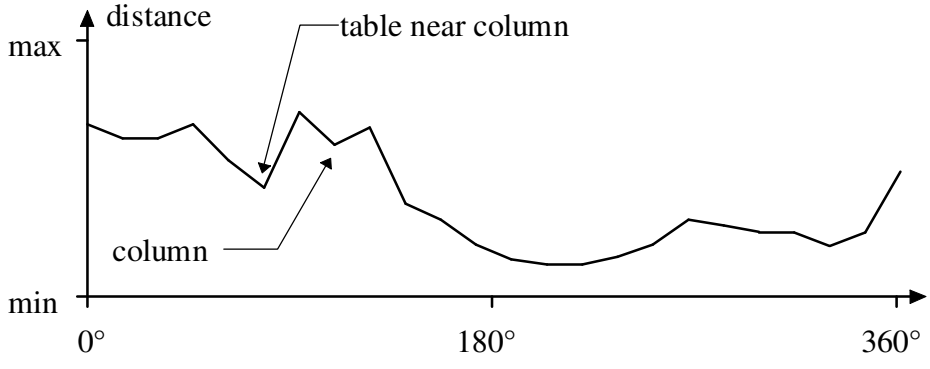

Figure 11: Second distance scan.