Image-based Homing
Using a Self-organizing Feature Map

Thomas Röfer
Center for Cognitive Sciences, University of Bremen
P.O. Box 330 440, D-28334 Bremen
E-mail: roefer@informatik.uni-bremen.de
Phone: +49 421 218 3495

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.

Figure 1: The robot

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 that is shown in Figure 1. This small robot has three wheels and is built with parts of a toy robot kit (Fischertechnik). It is able to drive in different directions without turning its body. A rotating photoreceptor is fixed on its top. It measures 64 values of brightness in one turn.

2 MATCHING IMAGES

Examples of the images that are taken by the brightness sensor are shown in Figure 2 and Figure 3. 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. It seems that the intensity in a small window is not sufficient to make correct correlations.

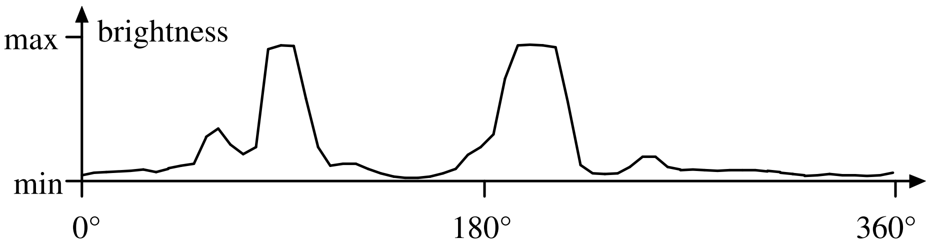

Figure 2: Typical image of a target position

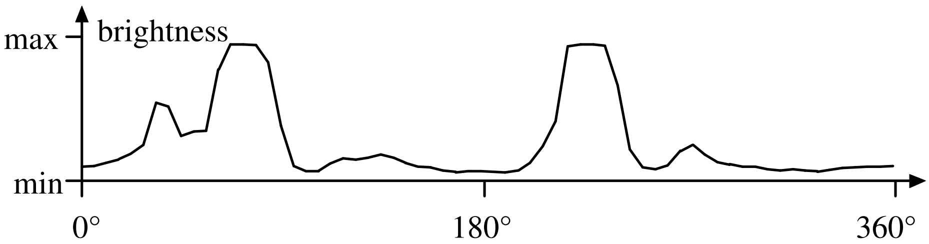

Figure 3: Image of a position 50 cm away from the target

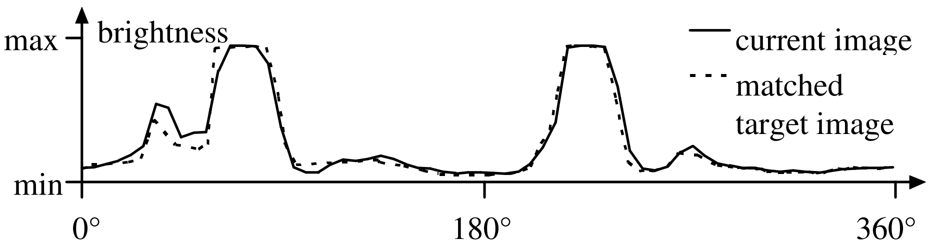

Figure 4: Matching of target image to current image

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.

3 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 component, 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: The the neighborhood and distance relations are those of a ring and the learning rule preserves the pixel's topology.

3.1 INITIALIZATION

The network is initialized and trained with vectors $\Psi_{\mathrm{I,i}}$. They consist of the pixel's index i devided by the total number of pixels, the pixel's brightness in the range of [0,1[and the first and second derivation of this brightness. The pixel's index is equivalent to the direction, in which this pixel is seen. The derivations are calculateted using a filter.

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

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

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

3.2 TRAINING OF THE NETWORK

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

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

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 $[best\text{-}\sigma, best+\sigma]$ 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 $v_{\mathrm{t,0}}$ - $w_{\mathrm{best,0}}$. 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.

$$\forall_{i \in [\text{best}-s_t,\, \text{best}]} \; w_{i,0}^{\text{new}} = v_{t,0} \dot{-} \frac{v_{t,0} \dot{-} w_{\text{best}-s,0}}{w_{\text{best},0} \dot{-} w_{\text{best}-s,0}} \left(w_{\text{best},0} \dot{-} w_{i,0} \right)$$
$$\forall_{i \in [\text{best}+1,\, \text{best}+s_t]} \; w_{i,0}^{\text{new}} = v_{t,0} \dot{+} \frac{w_{\text{best}+s,0} \dot{-} v_{t,0}}{w_{\text{best}+s,0} \dot{-} w_{\text{best},0}} \left(w_{i,0} \dot{-} w_{\text{best},0} \right) \tag{6}$$

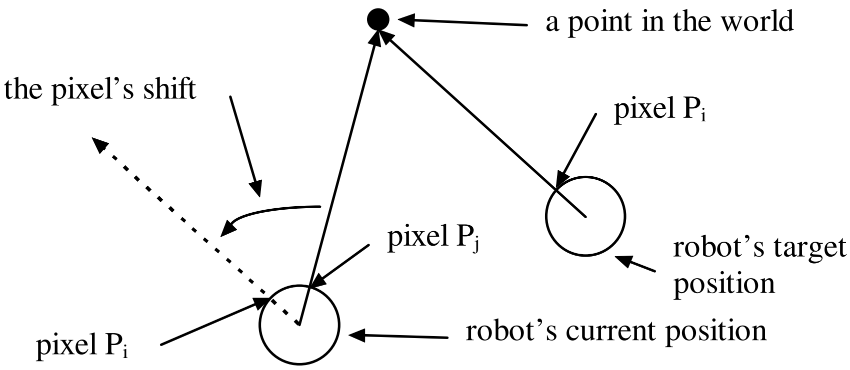

Figure 5: Pixel positions in the robot's current and target positions

4 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 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.

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:

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

4.1 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.

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, which can be used to calculate an adjusted shift s_i':

$$s_i' = s_i - \frac{1}{p} \sum_j s_j \tag{8}$$

4.2 CALCULATING THE DRIVING DIRECTION

Figure 6 illustrates the algorithm that calculates a homing vector for the shift of each pixel in the image.

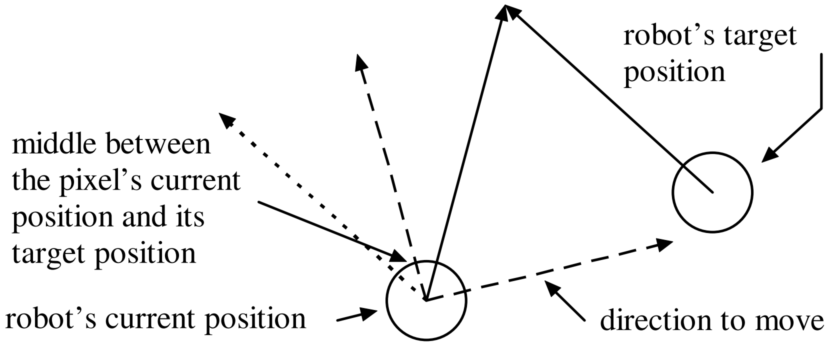

Figure 6: Calculating the homing direction

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 seen objects 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:

$$\beta = \tan^{-1} \frac{\sum_{i} s_i' \sin 2\pi\left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4}\right)}{\sum_{i} s_i' \cos 2\pi\left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4}\right)} \tag{9}$$

4.3 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 when 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 to make 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 4.2.

$$\gamma = \sqrt{\left(\sum_{i} s_i' \sin 2\pi\left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4}\right)\right)^2 + \left(\sum_{i} s_i' \cos 2\pi\left(w_{i,0} + \frac{s_i'}{2} + \frac{1}{4}\right)\right)^2} \tag{10}$$

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.

5 EXPERIMENTS

The robot was put at a random position and has taken an image. Then it was placed to another position, that was up to 1 meter apart. After the robot was started, it drove until the stop criterion was reached. 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. The robot always reached the target position with an error less than 3 cm. The robot needs around ten steps of "matching and moving" for a distance of one meter.

6 FUTURE WORK

There are three points of departure for future work:

- Image-based homing can be combined with an algorithm for obstacle avoidance. The homing module determines the direction to move, that may be modified by the avoidance module, if there is an obstacle. While the avoidance module controls movements the homing module is still tracking the homing direction. When the obstacle is passed the homing movements are continued.

- If the robot uses sequences of target positions it can move over long distances.

- If the image sensor would be able to recognize colors, the algorithm may be independent of different lightning situations. The color information can easily be added as further dimensions in the network's weight vectors.

7 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