Low-Level Visual Homing

Andrew Vardy and Franz Oppacher

Carleton University
School of Computer Science
Ottawa, Ontario, Canada
avardy@scs.carleton.ca
http://www.scs.carleton.ca/∼avardy

Abstract. We present a variant of the *snapshot model* [1] for insect visual homing. In this model a snapshot image is taken by an agent at the goal position. The disparity between current and snapshot images is subsequently used to guide the agent's return. A matrix of local low-level processing elements is applied here to compute this disparity and transform it into a motion vector. This scheme contrasts with other variants of the snapshot model which operate on one-dimensional images, generally taken as views from a synthetic or simplified real world setting. Our approach operates directly on two-dimensional images of the real world. Although this system is not a model of any known neural structure, it hopes to offer more biological plausibility than competing techniques because the processing applied is low-level, and because the information processed appears to be of the same sort that insects process. We present a comparison of results obtained on a set of real-world images.

1 Introduction

In [1] the *snapshot model* was proposed to explain the remarkable ability of honeybees to return to a place of interest such as a nest or food source after being displaced from that place. This model has attracted significant attention from researchers in the artificial life and artificial intelligence communities who have created computational models inspired by the snapshot model and implemented them both in simulation and in physical robots [4, 11, 3, 7, 14, 8–10]. All of these computational models operate on one-dimensional images. Even for the case of physical robots with cameras that generate two-dimensional images, the height dimension is almost immediately collapsed such that the image used for homing is effectively one-dimensional. Our approach differs from these in that we apply our processing directly on two-dimensional images. We find that the use of two-dimensional image features improves results over a similar model that operates only on one-dimensional features. Also, a number of these other models are tested either in simulated worlds or in simplified real world scenarios. We test our approach using the real world image album of [10] and compare our results with theirs.

In [8] a neural implementation of the snapshot model was presented. It showed that the processing necessary to implement the snapshot model could be achieved using simple neuron-like elements arranged in layers of interconnected rings. The outermost ring was exposed to a one-dimensional panoramic image of the homing agent's environment. Subsequent layers processed this image and extracted the necessary vectors to achieve homing. Our approach has been directly inspired by this work. It differs, however, in its use of two-dimensional images, and hence, two-dimensional processing layers. Also, the use of real-world test images prompted a number of modifications.

We can not state that our method is a model of any known neural circuit in an insect's brain. According to [9], *"...nothing is known so far about the neural circuits that realize visual homing in insect brains."* Still, there are three ways in which we claim that our model has some measure of biological plausibility. First is the fact that all computations are made via a matrix of locally-connected neuron-like elements. Secondly is the layered structure of our image processing matrix which preserves spatial relationships throughout each layer of processing. This is inspired by the retinotopic structure of the mammalian visual cortex where neurons on one layer seem to share receptive fields with the layers below them [5]. Our final claim to biological plausibility is in regard to the sort of information that our method employs. It operates on snapshots of a scene, stored in retinal coordinates. This is the primary contention of the snapshot model [1]. More recent work on wood ants [6] implicates the storage and use of two-dimensional templates for ant homing. We hope that our model can help to close the explanatory gap between a complete neural map of the hardware that implements visual homing in insects and a purely computational model that cares nothing for the details of implementation.

Our method operates by approximating, for each feature in the snapshot image, the direction in which that feature has moved in the current image. The features we use are image corners. The method used to determine the direction of feature movement is to examine the local gradient, created from features in the current image, at the position of the snapshot feature. Thus, we refer to our method here as the *Corner Gradient Snapshot Model*, or CGSM.

1.1 Snapshot-Based Homing

The snapshot model is originally due to Cartwright and Collett [1] who developed it to match data of honeybee search patterns. A model agent is placed at the goal and allowed to capture a snapshot image. It is then displaced and allowed a return attempt. The model operates on one-dimensional panoramic binary images. Features (landmarks) are defined as the centers of contiguous spans of all-on or all-off pixels. Features in the current image are paired with their closest matching counterparts in the snapshot image. Each pairing generates two vectors: one correcting for bearing and one correcting for differences in apparent size. The sum of all vectors yields the motion vector. One key requirement of the snapshot model is that the agent maintains a consistent orientation. There is some evidence to suggest that bees take on the same orientation when returning to a remembered place as they took when originally learning the layout of that place [2, 15, 6].

2 Processing Matrix

We will now describe the operation of the processing matrix for CGSM. The processing that is applied is depicted in figure 1. A summary of this processing follows: An input image is fed into the processing matrix. It is first smoothed and then corners are extracted as distinct features. If the agent is at the goal then the image of features is stored as the snapshot image. Gradients are formed around each feature and these gradients are locally compared with features in the snapshot image to generate vectors which indicate the direction that these features have moved in. These vectors that specify motion in the image are mapped onto vectors that specify the corresponding motion of the agent. Finally, these last set of vectors are summed to create the agent's overall motion vector. For the following layer descriptions, the image that is fed in from the previous layer is referred to as the *input image*. There is insufficient space to describe exactly how each layer's functionality can be distributed across a grid of low-level processing elements. We hope that the simplicity of each layer will provide the reader with assurance that this distribution is possible.

Gaussian Filter Convolves the input image with a 5x5 Gaussian mask.

Corner Extraction Extracts corners from the input image. The corner detection scheme we use is known as the Harris detector and is essentially the same as that presented in [13] (pages 82-85). The required computation is purely local in scope and requires only the application of some simple hard-coded algebra to the results of local convolutions and sums.

Local Maxima Extraction Applies thresholding and non-maxima suppression to find peaks in the input image [13].

Gradient Formation Forms decaying gradients around points (maxima) in the input image. These gradients allow the next layer to detect the direction in which a corner has moved from its idealized position at a stored snapshot point. Repeated gaussian filtering and diffusion are two possible ways of implementing this layer. We opt for a different approach which is quicker to compute with a serial computer. A single gradient mask is constructed which has a center value of unity and decays out radially according to $1/d^\alpha$, where d is the distance of the point in the mask from the mask center and α is an arbitrary constant ($0 > \alpha > 1$). Beginning with a zero image we place copies of this mask centered over every maxima point and take the maximum value for each output pixel from the values of all translated mask points aligned with that pixel.

Ring Operator Applied at every non-zero point in the snapshot image to generate a vector pointing in the direction that the feature has moved. This layer actually has two input images: the snapshot image, and the gradient image. At each non-zero point in the snapshot image, S, a ring of detector cells surrounding S is employed to detect the approximate direction of the gradient at that point. Ideally, this direction will point to the matching feature F. The vector from S to the detector cell with the highest response yields the uphill direction of the gradient.

Vector Mapping Maps image vectors into movement vectors. The next section provides more detail on this layer.

Vector Sum Sums all vectors in the input image to generate a single vector.

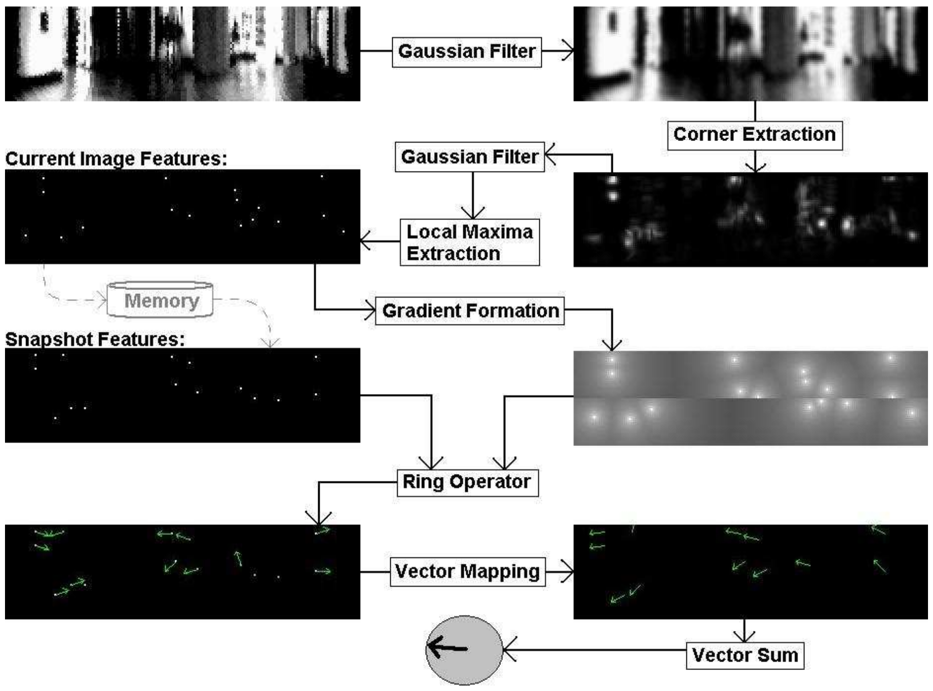


Fig. 1. Processing applied by the CGSM matrix. The input is the image in the upper-left corner. The output is the movement vector in the shaded circle at the bottom. Processing layers are shown as boxed text. Dashed arrows relate to the storage and recall of the snapshot image. The input image was taken from position (270,180) in the image album discussed in the *Results* section. This image was taken from a point directly to the right of the snapshot position (210, 180). Hence, the direction of the output movement vector is approximately correct.

2.1 Vector Mapping

The output from the *ring operator* layer is an image of vectors. If the agent were to move such that a feature F moved backwards along the vector attached to snapshot feature S then, assuming that F and S correspond and without considering any other features, the agent would have moved closer to home. The vector mapping layer's task is to translate image vectors into agent motion vectors.

We first illustrate the optic flow field that a moving agent with a panoramic imaging system would experience. Figure 2 shows such a flow field. In this instance, the agent is moving directly backwards and the focus-of-expansion (FOE) is consequently centred in the flow field at 180^o.

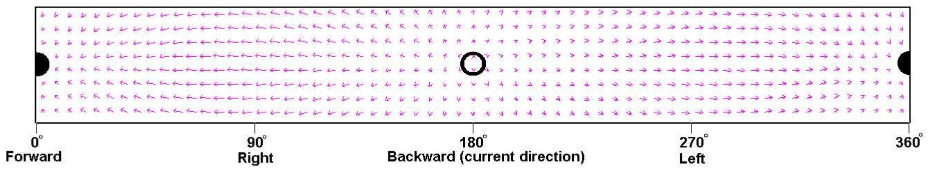


Fig. 2. Optic flow field for an agent moving directly backwards. The unfilled circle indicates the focus-of-expansion (FOE), while the filled circle indicates the focus-of-contraction (FOC). The angle on the horizontal axis indicates the viewing direction for that part of the image.

If the agent were translating in some other direction θ then the FOE would be at horizontal position θ and the whole flow field would be shifted accordingly. This is the key insight in developing the vector mapping layer. For each image vector we consider the corresponding snapshot feature's horizontal position θ. If the agent were to translate in the direction θ then the matching current image feature (presumed here to be nearby) would move purely vertically in the image. This is because the cross-section of the flow field that intersects the FOE has vectors with purely vertical components (see the cross-section of figure 2 at 180^o). Similarly, if the agent were to translate in the direction $\theta + 90^o$ then the matching current image feature would move purely horizontally (see the cross-section of figure 2 at $180^o + 90^o = 270^o$).

We will denote image vectors with lowercase letters and motion vectors with uppercase. We wish to map the image vector $\overrightarrow{v}$ into the agent motion vector $\overrightarrow{M}$. We first need to introduce some other vectors. The vector $\overrightarrow{I(\theta)}$ is the motion vector required to move the agent in direction θ. If the agent moves in this direction then the feature will move along the vertical image vector $\overrightarrow{i}$. The vector $\overrightarrow{E(\theta)}$ is the motion vector that would result if the agent moved in direction $\theta + 90^o$. The associated horizontal image vector is $\overrightarrow{e}$. The components of any motion vector is of the form [*amount to move forward*, *amount to move to the right*]. The components of any image vector is of the form $[x, y]$. The $\overrightarrow{I(\theta)}$, $\overrightarrow{E(\theta)}$, $\overrightarrow{i}$, and $\overrightarrow{e}$ vectors are as follows,

$$\overrightarrow{I(\theta)} = \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \qquad \overrightarrow{i} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$
$$\overrightarrow{E(\theta)} = \begin{bmatrix} \sin(\theta + 90^o) \\ \cos(\theta + 90^o) \end{bmatrix} \qquad \overrightarrow{e} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Note that if the position of $\overrightarrow{v}$ is in the bottom half of the image (which means that the snapshot feature S is in the bottom half of the image) then $\overrightarrow{i} = [0, 1]^T$.

For each snapshot feature S and the image vector at that position v, we first determine the combination of image vectors $\overrightarrow{i}$ and $\overrightarrow{e}$ that will sum to $\overrightarrow{v}$. That is we will solve $\overrightarrow{v} = a\overrightarrow{i} + b\overrightarrow{e}$ for the scalars a and b. Simple algebra yields that $a = \overrightarrow{v}.y/\overrightarrow{i}.y$, and $b = -\overrightarrow{v}.x + a\overrightarrow{i}.x$. To calculate $\overrightarrow{M}$ we just use the same combination of $\overrightarrow{I(\theta)}$ and $\overrightarrow{E(\theta)}$. That is, $\overrightarrow{M} = a\overrightarrow{I(\theta)} + b\overrightarrow{E(\theta)}$. θ itself is simply calculated: $\theta = 2\pi S.x/w$, where $S.x$ is the horizontal position of S in the image (also the horizontal position of $\overrightarrow{v}$ in the image), and w is the width of the image. Given a, b, and θ we can summarize the mapping from $\overrightarrow{v}$ to $\overrightarrow{M}$ as,

$$\overrightarrow{M} = \begin{bmatrix} a\sin\theta + b\sin(\theta + 90^o) \\ a\cos\theta + b\cos(\theta + 90^o) \end{bmatrix}$$

3 Results

We compare the homing performance of CGSM against an implementation of the average landmark vector model which was designed to operate on real-world images [10, 12]. The average landmark vector model can be considered a variant of the snapshot model [9]. This implementation describes how a 1-D image is taken from a 2-D panoramic image and how features of this image are subsequently extracted. Once the features have been extracted the average landmark vector model can be applied in its original form [9]. To obtain the 1-D image a horizontal strip is extracted from the 2-D image. The columns of this strip are averaged to produce a 1-D image. The image is then low-pass filtered. If the image is to be used as the snapshot image than an adaptive threshold value is also extracted which will be stored along with the average landmark vector. For images that are used for homing the points where the smoothed 1-D image crosses the threshold line are extracted as features. We shall refer to this implementation of the average landmark vector model as *XALV* (eXtracted Average Landmark Vector).

The authors of [10] have very kindly made their images available to us and we use them here to compare our results with theirs. These images were taken from a grid of positions spaced 30 cm apart within a $9m \times 3m$ area. The environment was an unmodified entrance hall of a university building. These pictures were taken by a robot-mounted camera pointed upwards at a conical mirror. The robot's orientation was kept constant throughout the image capturing process. We have taken this album of 300 images and unfolded all of them into rectangular panoramic images, 180×48 in size. Figure 3 shows an example image from the album and an unfolded version of the same image.

Figure 4 shows the homing performance of both the XALV and CGSM models on the image dataset using the image taken at position (210,180) as the snapshot image. This image is depicted in figure 3(b). Both models calculate home vectors for all images. These home vectors are depicted in figure 4. The figure also shows the success of homing for all grid positions. A pure white cell indicates successful homing. Homing is considered successful if a path along the home vectors exists from the position in question to the home position. We calculate a quantity called the *approach index* for each position,

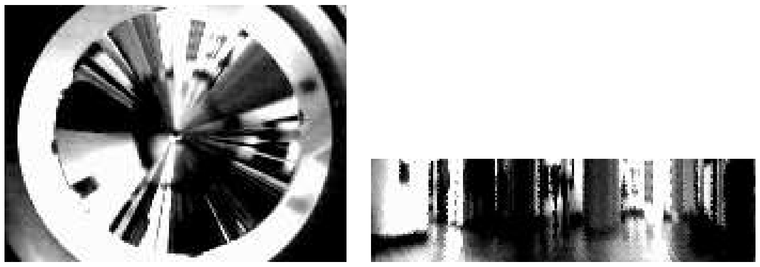

(a) Camera image (b) Unfolded image

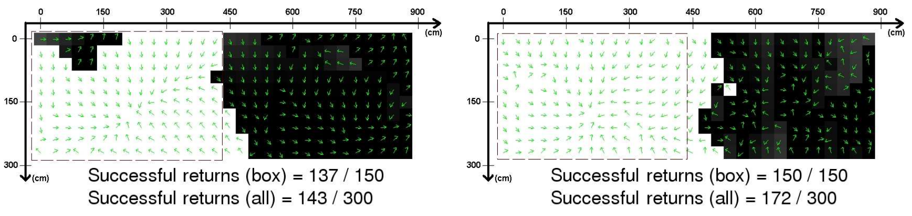
Fig. 3. The image unfolding process.

$$\text{approach index} = \frac{\text{starting distance} - \text{closest distance}}{\text{starting distance}}$$

The quantity *starting distance* is the euclidean distance from the starting position to the home position. *Closest distance* is the smallest distance from the current position to the home position along the path from the starting position. The approach index for a successful starting position is 1. If, however, the path from the starting position leads away or perpendicular to the home position then the approach index will be 0. If the path approaches the home position but does not reach it then the approach index will be between 0 and 1. The purpose of the approach index is to judge the success of homing even if it is not completely successful. The level of whiteness shown in figure 4 is proportional to the approach index for the corresponding position.

Figure 4 has a dashed box surrounding the left half of the dataset. This box encloses the positions used to generate the results presented in [10]. It had been our intention to replicate these results precisely, however we have not been able to reconstruct all of the necessary parameters. This being said, our re-implementation of the XALV model seems to achieve better results. The original result for positions within the dashed box was that successful homing could be achieved from 98 of the 150 grid positions. Our re-implementation achieves successful homing from 137 of these 150 positions. This improvement is eclipsed by CGSM which achieves perfect homing (150 out of 150) within the dashed box. Considering the whole of the dataset, our re-implementation of XALV achieves 143 / 300, while CGSM achieves 172 / 300. Both methods do relatively poorly in that part of the grid furthest from the home position.

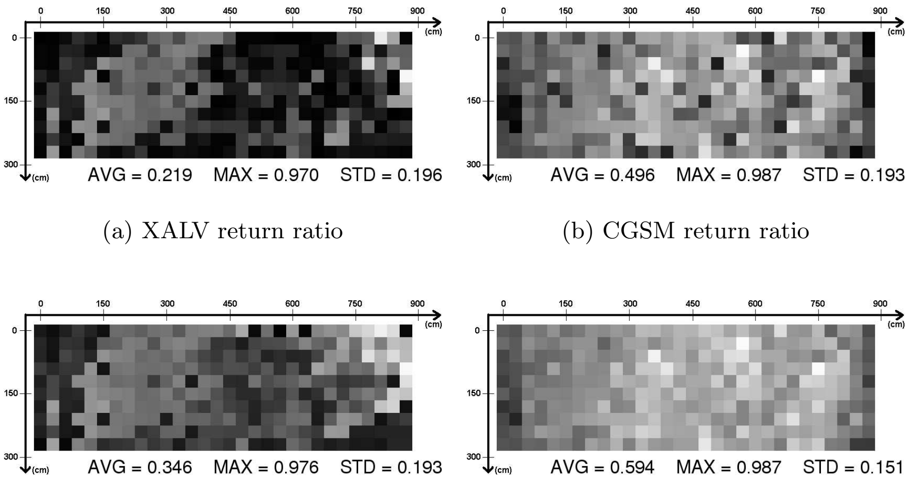
The success of homing to one particular goal position does not tell the whole story. It may be that one method achieves good results from certain goal positions but not from others. To investigate this issue we have generated 300 homing maps for the 300 image dataset using each image of the dataset, in turn, as



(a) XALV home vectors (b) CGSM home vectors

Fig. 4. Homing maps of the XALV and CGSM models for goal position (210,180). Home vectors are shown for images taken at corresponding positions. The whiteness of each position is proportional to that position's approach index. Positions within the dashed box area were used for the results presented in [10].

the home position. For each homing map we count the number of successful homing positions and divide by 300. This is the *return ratio*. Figure 5 shows the return ratio for all images as the level of whiteness of each cell. This figure also shows the approach index ratio, which is calculated similarly to the return ratio except that the approach index is used as opposed to the number of completely successful returns.



(a) XALV return ratio (b) CGSM return ratio

(c) XALV approach index ratio (d) CGSM approach index ratio

Fig. 5. Return ratio and approach index maps for XALV and CGSM.

The average and maximum return ratios are higher for CGSM than for XALV. The average and maximum approach index ratios are also higher. In addition, the standard deviation for these ratios are both lower for CGSM. For 256 of the 300 homing maps the return ratio is strictly higher for CGSM than for XALV. For 263 of these homing maps the approach index ratio is also strictly higher.

The area that surrounds a goal position from which successful homing can be achieved is known as the *catchment area*. We can translate the return ratio into squared metres to give an idea of the size of the catchment area for these two models. The average catchment area for CGSM is 13.4 m^2 (max. 26.6 m^2) while the average area for XALV is 5.9 m^2 (max. 26.2 m^2).

3.1 Discussion

Some caveats should first be mentioned in interpreting the results presented above. Firstly, all images were taken with constant orientation. This is an unrealistic scenario for actual robots homing in the real world. Even for robots with sophisticated odometry and compass systems, errors in orientation are inevitable. Secondly, it was reported in [10] that changing lighting conditions seriously impacted on the homing performance of XALV. Thus, care must be taken in citing the catchment areas above as if they referred to actual environment-independent measures. This being said, it may also be true that the catchment area for this environment is *larger* than quoted above. If a set of images covering a larger area was employed it might be found that the catchment area would extend out further. Indeed in figure 4(b) the catchment area is prematurely bounded in places by the edges of the grid.

Our results show superior performance for CGSM over XALV. Both of these models can be considered variants of the snapshot model. We take the superior performance of CGSM to mean that snapshot-based homing models can do better by utilizing information that is implicitly encoded in both dimensions of an image seen by a homing agent. Further, we have shown that the success of a biologically plausible model tested only in simulation [8] can be transferred into a more realistic test environment where homing is based upon real-world images.

4 Conclusions

We have presented a variant of the snapshot model which is capable of successful homing using real-world test images. It exhibits improved performance over a model which did not make use of both input image dimensions. The structure and nature of our processing matrix was inspired by real biological systems.

There are a number of directions in which to take this work. One clear possibility is to test out our ideas on an actual free-roving robotic platform and assess how well CGSM performs in various environments. Analysis of error conditions is also of prime importance. Currently, we are assessing the usability of features other than corners, and trying to determine whether there are methods other than our gradient-based one to determine in which direction a feature has moved using only low-level computations.

Acknowledgments

Thanks to Matjaz Jogan. Particular thanks to Thorsten Roggendorf and Ralf Möller for making their images available and assisting with the replication of their results. This work has been partially supported by NSERC Postgraduate Scholarship B - 232621 - 2000.

References

1. B. A. Cartwright and T.S. Collett. Landmark learning in bees. *Journal of Comparative Physiology*, 151:521–543, 1983.
2. T.S. Collett and J. Baron. Biological compasses and the coordinate frame of landmark memories in honeybees. *Nature*, 368:137–140, 1994.
3. Matthias O. Franz, Bernhard Schölkopf, Hanspeter A. Mallot, and Heinrich H. Bülthoff. Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, 79:191–202, 1998.
4. J. Hong, X. Tan, B. Pinette, R. Weiss, and E.M. Riseman. Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacremento, CA*, pages 620–625, 1991.
5. D. H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.
6. S.P.D. Judd and T.S. Collett. Multiple stored views and landmark guidance in ants. *Nature*, 392:710–714, 1998.
7. D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems, Special Issue: Biomimetic Robots*, 1999.
8. R. Möller, Marinus Marus, and D. Lambrinos. A neural model of landmark navigation in insects. Technical Report IFI-AI-99.19, Artificial Intelligence Lab, University of Zurich, 1999.
9. Ralf Möller. Insect visual homing strategies in a robot with analog processing. *Biological Cybernetics*, 83(3):231–243, 2000.
10. Ralf Möller, Dimitrios Lambrinos, Thorsten Roggendorf, Rolf Pfeifer, and Rüdiger Wehner. Insect strategies of visual homing in mobile robots. In B. Webb and T. Consi, editors, *Biorobotics - Methods and Applications*. AAAI Press / MIT Press, 2001.
11. Thomas Röfer. Controlling a whellchair with image-based homing. In *Proceedings of AISB Workshop on Spatial Reasoning in Mobile Robots and Animals, Manchester, UK*, 1997.
12. Thorsten Roggendorf. Visuelle landmarkennavigation in einer natürlichen, komplexen umgebung. Master's thesis, Universität Bielefeld, 2000.
13. E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
14. Keven Weber, Svetha Venkatesh, and Mandyam Srinivasan. Insect-inspired robotic homing. *Adaptive Behavior*, 7:65–97, 1999.
15. J. Zeil, A. Kelber, and R. Voss. Structure and function of learning flights in bees and wasps. *Journal of Experimental Biology*, 199:245–252, 1996.