

Learning View Graphs for Robot Navigation

Matthias O. Franz Bernhard Schölkopf
Philipp Georg Hanspeter A. Mallot Heinrich H. Bülthoff
Max-Planck-Institut für biologische Kybernetik
Spemannstraße 38
72076 Tübingen
Germany
franz/bs/pg/ham/hhb@mpik-tuebingen.mpg.de

Abstract

We present a purely vision-based scheme for learning a parsimonious representation of an open environment. Using simple exploration behaviours, our system constructs a graph of appropriately chosen views. To navigate between views connected in the graph, we employ a homing strategy inspired by findings of insect ethology. Simulations and robot experiments demonstrate the feasibility of the proposed approach.

Introduction¹

To survive in unpredictable and sometimes hostile environments animals have developed powerful strategies to find back to their shelter or to a previously visited food source. Successful navigation can already be achieved using simple mechanisms such as association of landmarks with movements (Wehner et al. 1996) or tracking of environmental features (Collett 1996). To understand more complex forms of spatial behaviour like finding shortcuts, however, we have to go beyond reactive control strategies, towards systems with internal states. In as far as they support navigation behaviours, these states can be thought of as representing certain task-relevant spatial aspects of an animal's environment.

The spatial representations that have been proposed in the literature — sometimes referred to as *cognitive maps* — differ in the type of sensory input being utilized and in their geometric power. Regarding the latter aspect, a main distinction has been the one be-

tween metric maps and topological maps. Traditionally, robotics approaches have focused on constructing accurate global metric representations, based on a variety of mostly non-visual sensors. Besides the high computational costs the problem with such geometric

models is that they tend to contain a large amount of irrelevant information while the visual or non-visual cues that lead to their construction are not specifically represented. The goal of any representation, however, is to support navigation tasks, thus researchers have started to explore possibilities of solving these tasks with more parsimonious representations and less complex algorithms (e.g. Kuipers and Byun, 1991, Mataric, 1991, Bacheider and Waxman, 1995, or, for biological behaviours, Braitenberg, 1984). For instance, Schölkopf and Mallot (1995) have shown that learning a graph of views and movement decisions is sufficient to generate various forms of navigation behaviour found in rodents, such as finding previously visited locations, or updating spatial representations with egomotion information to maintain knowledge about one's position even without external information. This scheme has subsequently been implemented on a mobile robot (Mallot et al., 1995). The present study undertakes to extend this approach from mazes to open environments. To this end, it was necessary to generalize both the type of spatial representation used and the mechanisms for constructing and utilizing this representation. These points shall be explained in the next section, followed by a description of an experimental investigation with simulations and robot implementations. We conclude our study with a discussion of the results.

Navigation in Open Environments Discrete Representation of Continuous Space

In view-based navigation, we use visual information to guide navigation in a 2-dimensional space. The reason why this is feasible at all is the fact that there is a continuous mapping between position space (which could contain also the direction of view) and the space of all possible views: for each spatial position, a certain view

¹Copyright ©1997 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept, ACM Inc., fax +1 (212) 869-0481, or permissions@acm.org.

is perceived, and this view changes continuously as the observer moves around in space.² Unfortunately, this mapping can be singular, as identical views can occur at different spatial locations, i.e. there need not be a global coordinate system (of spatial positions) on the manifold of all possible views (cf. Fig. 1). In principle, this can be dealt with using history information: in points with ambiguous views, we can use the information that the observer moves continuously, and hence choose a location which is consistent with the last position. Complete knowledge of this manifold would thus be a very powerful means of determining one’s spatial position. Memory and computation requirements, however, prohibit storing everything. Moreover, if we are not interested in determining positions at arbitrary times, but rather in carrying out specific navigation tasks, as for instance path planning, this is not at all necessary. In that case, it could be sufficient to store views which allow the description of relevant paths. Considerations along these lines lead to a weaker representation of the view manifold, namely by a graph of representative views and connections.

Graph-based representations of space have been employed in a number of studies (McNaughton, 1989; for reviews, see O’Keefe, 1991, Gallistel, 1990). We choose as our starting point the scheme of Schölkopf and Mallot (1995), which consists of a graph of views at the junctions of a maze. In discretized environments like mazes, there is a canonical set of views to store: since no movement decisions need to be taken while traversing corridors, the views necessary to support path planning are solely those at junctions. In generalizing the view graph scheme to the case of open environments, we have to find a set of views which are representative for the manifold (in the following referred to as *snapshots*), with connections between them. The resulting graph connectivity is subject to the following trade-off: each connection which is added allows the representation to get closer to the original manifold, however if the connections cannot be travelled along reliably, the graph becomes a poor means of path planning.

Finally, if we restrict ourselves to the use of purely topological information, excluding e.g., to label the graph connections with directions, it is imperative to use a method by which we are able to find views which are connected to a given start view. Following the usage in the literature, we will refer to such a method as *homing*.

In the next two sections, we describe the two crucial components of our scheme; namely the procedures for taking snapshots and for homing. Following that, we shall explain how these components can be combined into graph learning strategies.

²If the views are vectors where each component was recorded with a Gaussian receptive field, the view will be a smooth function of the position.

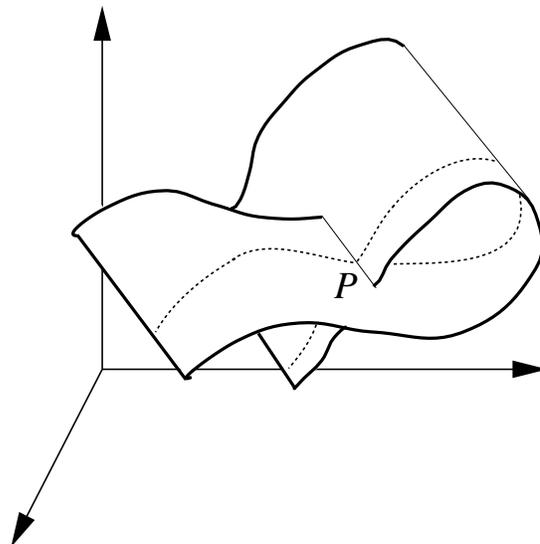


Figure 1: Caricature of the view manifold, consisting of all views that can be seen by a continuously moving observer. The manifold is embedded in a Euclidean space whose dimensionality is the number of camera pixels. The actual structure of the manifold is much more complicated, with holes caused by obstacles, and including a tubular structure due to the possibility to take snapshots at all directions between 0° and 360° . Point P marks a singularity of the coordinate system inherited from position space: e.g., if one moves along the dotted path, the same view occurs twice at different spatial locations. Similarly, there could be regions in position space where the visual input locally does not change, leading to another type of coordinate singularity (not depicted).

Sampling the View Manifold with Snapshots

The nodes of the view graph are identified with snapshots of the surrounding panorama, namely 360° records of the horizontal grey value distribution. This has several advantages for homing and place recognition:

1. If a robot translates on a planar surface, then the poles of expansion and contraction of the generated flow field are on the horizon. All objects lying on a great circle through the poles of the flow field will not leave the circle during translatory movement. Apart from the possibility of occlusion, this means that they remain visible in the 360° sensor during motion.
2. Similarly, circles perpendicular to the axis of rotation are invariant under rotations: if the robot rotates on a flat surface, objects do not change their altitude in the visual field. As the horizon is a great circle perpendicular to the above axis of rotation, we

also have invariance under translation, thus objects cannot leave it by arbitrary robot movements in the plane.

3. If the center of the sensor ring is placed at the rotation axis, the rotational and translational flow field can easily be separated: Rotation causes a uniform flow field on the sensor ring which can easily be determined and subtracted to obtain the translational flow field.

These considerations illustrate that disadvantages resulting from analyzing just a small image region can in some respects be outweighed by special properties of a carefully chosen subregion.

Ideally, the set of snapshots taken to represent a given environment should satisfy two criteria: first, the views should be distinguishable — in purely graph-based maps, stipulating that the graph nodes be distinguishable is the only way to guarantee the possibility of navigation to specific views. Second, the distance of neighbouring views in position space should be small enough to allow reliable navigation between them. In the present study, we adapt the spacing of the snapshots to the rate of change of the incoming images by imposing a minimum image distance on the snapshots taken. Whenever this minimum distance is exceeded by the presently perceived view, a new snapshot is taken.

More general, this can be viewed as a pattern classification problem. To solve this problem, we used learning machines which we trained on data generated by the robot during exploration. The classifier has to detect whether a given view belongs to a location that is closer than a given distance expressed by a time delay t_d (the robot moves with constant velocity). To construct training examples the robot carries out an exploration consisting of straight movements with lengths uniformly drawn from $[0, 2t_d]$. After each line segment, a snapshot is taken and a new movement direction is chosen at random. The two classes of training examples then comprise view pairs with a connecting segment line shorter or longer than the equivalent of t_d , respectively.³ This approach thus connects the two different metrics which can be found on the view manifold: we infer spatial closeness of views from their degree of similarity.

After training, we have a classifier which, for a given environment, is able to detect whether views are (spatially) distant enough. Clearly, the same classifier can also be used to detect proximity — we shall come back to this point below, when we describe methods of learning short-cut connections in graphs.

³Clearly, this simplistic strategies could be improved: we are free to tailor the exploration to the needs of the classifier that we want to train (*active learning*). Moreover, there are two obvious ways of preprocessing the input data; first, one can use the difference image as the input to our classifiers, second, one can even reduce the input to the Euclidean distance of the two views.

For an experiment, we generated 5000 32-dimensional training examples in a simulated environment (Fig. 7), and another 5000 test examples, recorded in the same exploration *after* the training examples. As classifier input, we used the Euclidean distance of the two view vectors, after first rotating one of them such as to maximize the overlap with the other one. A simple thresholding classifier then achieved classification error rates of 15%. Fig. 2 shows the distributions of view distances (after rotation) for the two classes.

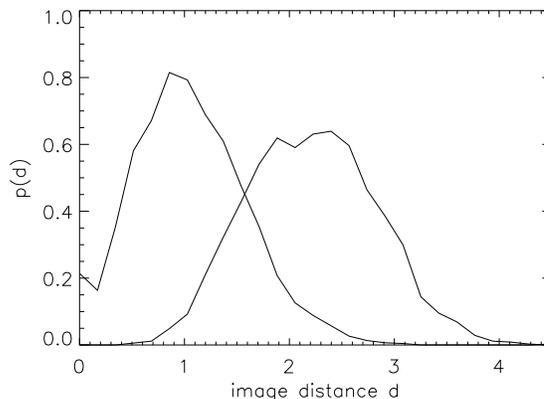


Figure 2: View distance distribution of spatially close (left curve) and distant (right curve) views. We used 5000 examples generated in the environment in Fig. 7 (see text). The spatial distance threshold defining the two classes was set to $1/10$ of the arena’s diagonal. The images were vectors in $[-1, 1]^{32}$.

Navigating between Places: View-based Homing

Due to their limited information processing capabilities, insects use very effective optical navigation methods which require only modest computational efforts. This makes them a natural prototype for robotic implementations, where computational costs often prohibit implementation of more complex visual algorithms. The homing method described in this section is based on two ideas taken from insect biology: first, the so-called snapshot theory and second, matched filters. The snapshot theory has been developed by Cartwright and Collett (1987) to explain the search behaviour of honey bees which enables them to relocate a food source using a snapshot of the surrounding scenery taken during a previous visit. The direction of the food source after a displacement can be inferred from the actual view by comparing it to the snapshot: image regions in the direction of the displacement are expanded while the image in the direction of the food source is contracted.

Cartwright and Collett propose to compare the relative bearings of nearby objects for this task. This poses

some problems from an algorithmic point of view: objects have to be separated from the background and the correspondence of their images in the snapshot and the actual view must be established. One need not solve these problems explicitly if one uses matched filters incorporating prior knowledge about expected displacements. For instance, to detect self-motion during flight, the blowfly *Calliphora* uses a matched neural filter coding for optical flow patterns which are similar to retinal motion fields experienced during translation or rotation (Krapp, & Hengstenberg 1996). The degree of match between the predefined flow field and the currently perceived field determines the strength of the filter signal which can be used to correct the deviations.

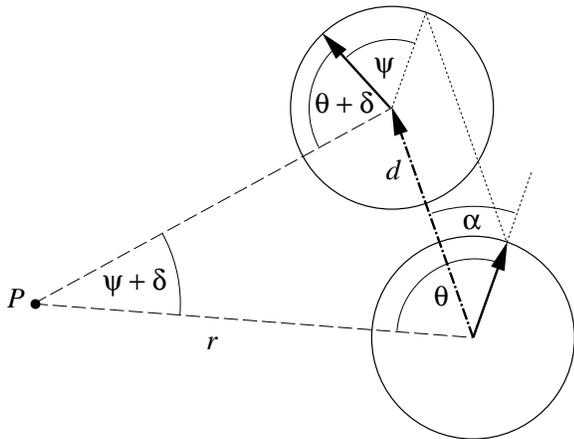


Figure 3: Change δ of the viewing angle of point P after displacing a ring sensor at a distance d in direction α and rotating it by ψ

This idea can also be used in the above optical homing method. Assume we have a sensor ring which allows to record a 360° view of the surrounding panorama. The position of an image point on the sensor ring is denoted by the angle θ . We displace the sensor ring in direction α by a distance d and change its orientation by the angle ψ . The image of Point P at distance r is shifted from θ to a new position $\theta + \delta$. For the triangle in Fig.3 we obtain the relation

$$\frac{r}{d} = \frac{\sin(\theta - \alpha + \psi + \delta)}{\sin(\psi + \delta)} \quad (1)$$

which implies

$$\tan(\psi + \delta) = \frac{d \sin(\theta - \alpha)}{r - d \cos(\theta - \alpha)}. \quad (2)$$

Let R be the average distance of all points that generate the image, and write $r = R + r'$.⁴ Eq. (2) now has

⁴Note, that no assumptions have been made regarding the distance of the points after the movement.

the form

$$\tan(\psi + \delta) = \frac{\frac{d}{R} \sin(\theta - \alpha)}{1 + \frac{r'}{R} - \frac{d}{R} \cos(\theta - \alpha)}. \quad (3)$$

For $r' \ll R$ and $d \ll R$ Eq. (3) becomes independent of r'

$$\tan(\psi + \delta) \approx \frac{d}{R} \sin(\theta - \alpha), \quad (4)$$

and therefore independent of the specific object distances. The approximation holds if the visible objects do not differ largely in distance and the robot's movement is small compared to the object distances.

The flow field on the ring generated by Eq. (4) can be used as a matched filter which has to be matched in the three parameters ψ , α and d/R . This is done the following way: for all parameter values the current view is warped using Eq. (4) and subsequently compared to the snapshot at the home position. The value of α obtained from the best match gives an estimate of the home direction.

The change of object azimuth after displacement has been used by several groups for homing tasks. Hong et al. (1991) identified and matched image features in panoramic images with constant orientation. They used this scheme to successfully guide a mobile robot along a corridor. The scheme of Basri and Rivlin (1995) linearly interpolates between image features in model views and infers home direction from the interpolation coefficients, but works only from a fixed starting position. Wittmann (1995) calculated the flow field using a correlation scheme on a resolution pyramid. In Röfer's approach (1995) a Kohonen network had to learn the correspondence between snapshot and momentary view.

The matching of three parameters requires relatively small computational resources compared to other methods for the calculation of optic flow. On an SGI Indy workstation, the calculation of a home vector from 78-dimensional views took below 40 ms which allows real time image processing at video rate. The continuous home vector calculation 'on the run', results in smooth trajectories to the home position. The speed of this method could be further improved by estimating the parameters ψ and d/R independently from image data, but this topic will be addressed elsewhere.

The applicability of the approximation in Eq. (4) was tested first in virtual reality simulations using indoor and outdoor scenes (cf. Fig. 8). Fig. 4 shows the results for different distances d . To evaluate the accuracy of our method we use the average homeward component of the estimated home vector. As long as the value stays above zero, the robot moves on the average nearer to the goal marked by the snapshot. As a practical definition for the radius of the snapshot's catchment area, we take the distance where the homeward component falls below 0.5. This means that inside this area the average radial component towards the goal is larger than the tangential component. As can

be seen from Fig.4 the actual catchment area is much larger than could be expected from Eq. (3). This can be explained by the fact that the sign of the displacement stays the same also for larger values of d/R , so that the linear approximation of the best match still gives good results beyond its mathematical limit. Experiments on real robots showed that the robustness of the scheme persisted under noisy real world conditions imposed by camera and video transmission. The high update rate and the symmetry properties are able to cancel the noise effects on the homing performance to a high degree. Statistical data from experiments with real robots are currently being gathered.

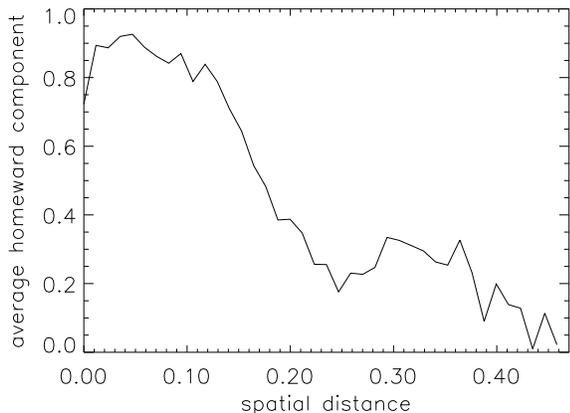


Figure 4: Average homeward component, for views belonging to 10000 pairs of random locations, obtained from Eq. (4) in the environment in Fig. 7. Distances are measured relative to the arena’s diagonal.

The accuracy with which a point can be reached depends mainly on the angular resolution of the sensor ring, namely the number of sensor elements and their receptive fields. A displacement can only be detected if it generates a sufficient angular shift. In Fig. 4 the sharp decrease of the homeward component marks the distance from which the location cannot be approached further. The point at which the home vectors generated by the scheme become inconsistent can be used to decide if the goal has been reached.

Graph Learning

In order to learn view graphs, the two procedures described above for taking snapshots and homing towards them have to be complemented by additional building blocks which will be described in the following (cf. Fig. 5).

Route learning. As explained before, the system uses a classifier to decide whether all recorded nodes are sufficiently distant. If this condition is fulfilled, the robot

takes a new snapshot and links it to the last one. This route learning procedure has no way of forming new connections to previously visited views, i.e. the resulting graphs will be mere chains. By adding the following simple behaviour, we can get nontrivial graphs:

Link verification. Whenever one of the previously stored views is found to be spatially close to the current view, and the connection between the last snapshot and the one found to be close has not yet been learnt, we decide to home to the latter one. If homing is successful, we include the newly learnt connection into the graph. Deciding whether a previously stored view is spatially close to the current view again is a pattern classification problem which can be solved with the methods described above. Thus, the classifier has two tasks in our system: to decide when to take snapshots and to detect candidates for shortcuts between the chains of the graph.

If an already linked view is found to be in the direction of the next exploration step the system homes to it as well. This procedure does not produce additional knowledge, but has the effect that links intersecting previously stored links are less likely to be recorded. Link verification could in principle also be used for the links learnt by the route learning procedure. For reasons of excessive exploration times, we did not resort to this more cautious strategy.

Emergency behaviours. Distance sensors and bumpers, with low-level escape behaviours, are used to keep the robot away from obstacles. In cases where the robot gets lost (e.g. if a verification was not successful), we start a new graph, which will typically get connected to the old one in due course. Areas where bumpers or proximity sensors detect objects show a high rate of change of the optical input due to large image motion caused by nearby objects. Exploration of these areas thus requires a large number of snapshots which would ultimately lead to a fractal graph structure near objects. To prevent the navigation system from getting ineffective, the robot is not allowed to take new snapshots if nearby objects are detected. The resulting graph structure tends to concentrate in open space rather than around obstacles.

Local Exploration Strategies for Graph Learning

The basic elements of our graph learning system have been introduced in the last section, however, they do not fully determine *exploration strategies*. E.g. nothing has been said about how to choose a new movement direction after a snapshot has been taken. The strategies that we will present in the following have been motivated by the principle of *maximizing knowledge gain* (Thrun, 1995). As we have not formalized any notion of knowledge, this principle was used as a qual-

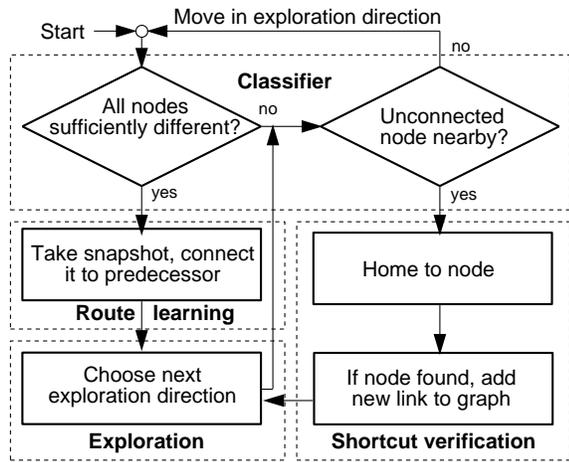


Figure 5: Short sketch of the graph learning algorithm used.

itative guideline. In the context of spatial representation, knowledge gain is possible for instance through the recording of new links and new snapshots.⁵ Clearly, a high number of stored snapshots is desirable in order to represent the environment accurately, on the other hand, if they are not sufficiently connected by links, they are not useful for path planning. In addition, the distribution of the snapshots is equally important as their number. We have tested a variety of rules and behaviours in an attempt to ensure that the resulting graph contains as much knowledge as possible.

In the following, we shall describe these rules, starting from rules which apply to general exploration and search problems. In our case, these concern primarily the choice of the next direction to explore after a snapshot has been taken.

Brownian exploration. The simplest conceivable rule is just to choose a random direction and then to go straight until the next snapshot. The resulting Brownian motion pattern has the advantage that eventually every accessible point of the environment will be explored without the danger that the exploring agent is caught in an infinite loop. This strategy does not use any knowledge about the environment and thus provides the baseline against which other search strategies have to be compared.

Kinesis. A straightforward way to include optical information in a Brownian search is to adapt the turning angle to the rate of change of the optical input. If for

⁵As knowledge, we count everything which contributes to making the map more useful for solving navigation tasks like path planning (path planning on the graph can be done with a variety of algorithms, e.g. as by Schölkopf and Malot, 1995).

instance the average turning angle or, as in the present system, the turning frequency is increased when the optical input changes rapidly, then areas which have to be covered by a denser net of snapshots due to a rapid change of views are also explored more thoroughly.

Fixed turning angles. Good results have also been achieved with fixed turning angles in combination with the rules described later in this section. In general, smaller angles lead to a faster coverage of the whole accessible area, larger angles to a more thorough exploration.

The following exploration features makes use of the local properties of the edges and nodes of the graph being constructed.

Exploration of the largest open angle. Our navigation scheme is designed such that all nodes of the view graph remain in the catchment areas of their respective neighbours. This property can be used to choose the next exploration direction, if a node has already more than one link. The system determines the home vectors to all neighbouring nodes and directs the next exploration step in the largest open angle.

Limiting the connectivity of nodes The effectivity of exploration can be increased by limiting the number of links a node can have. Similarly, the largest open angle can also be used to determine whether a node has been fully explored. If the largest open angle is smaller than a preset value, or the number of links is higher than a threshold, the system can go on to other nodes. Using this strategy, exploration tends to diffuse to less explored nodes and areas.

Non-links. The graph produced by the navigation system so far includes no information about failed actions as e.g. obstacle encounters during exploration or failed verification of shortcuts. The exploration can be made far more effective by memorizing failed actions as “non-links” thus preventing them from being repeated. This is also a way of including information about obstacles in the graph structure.

In this study we were only interested in evaluating the performance of local rules, but the approach can easily be extended to include global rules such as searching the graph for less explored nodes, or deleting unnecessary links.

Fig. 6 shows experimental data illustrating some properties of our scheme. In the simulated exploration, we first observe an increase in the number of both graph nodes and links; however after some time almost no new snapshots are taken — the view manifold has been sufficiently densely sampled, while the verification procedure still adds new links to the graph. Fig. 7 shows an example of a learnt view graph. To

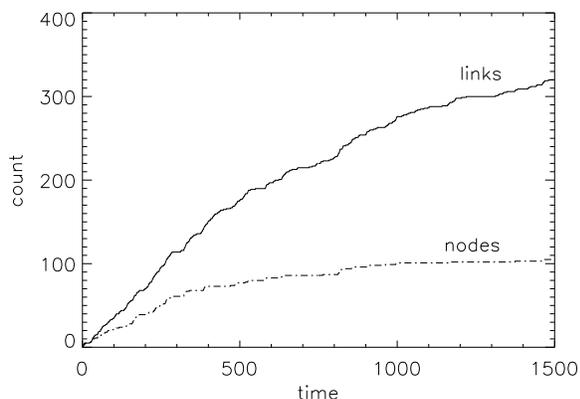


Figure 6: Numbers of graph nodes and links as functions of exploration time during a typical exploration run. It can be seen that due to filling in of short-cut links, the number of links still increases when the number of nodes has almost reached its final size. The diagonal of the arena (Fig. 7) can be traversed in approximately 30 time units.

assess the reliability of the acquired view graph, we tested whether the simulated robot was able to reproduce the recorded links by homing. Out of all graph links, the success rate was always between 94% and 98%.

Implementation

Simulations

Virtual reality. The navigation system described in the previous sections was developed and tested in simulated environments before it was exposed to real world conditions on a robot platform. Since the algorithms are completely view driven we put special emphasis on the creation of realistic image sequences. For this purpose we used the virtual reality software package “SGI PerformerTM” on a high performance graphics computer (SGI Onyx Reality Engine 2). The navigation system controlled the movements of a virtual agent in various indoor and outdoor scenarios as shown in Fig. 8. Training data for classifiers and evaluation data for the homing scheme can be obtained in relatively short time from simulation runs in this type of virtual environment.

Two-dimensional world. For the evaluation of the exploration strategies, realistic view conditions play only a minor role. Therefore, we simulated very simple two-dimensional models of the kind shown in Fig. 7. Views were obtained using standard ray-tracing techniques. Since the thorough exploration of a non-trivial environment is very time consuming, the computational simplicity of these models allowed us to test the different strategies in a variety of different environments with complex topology.

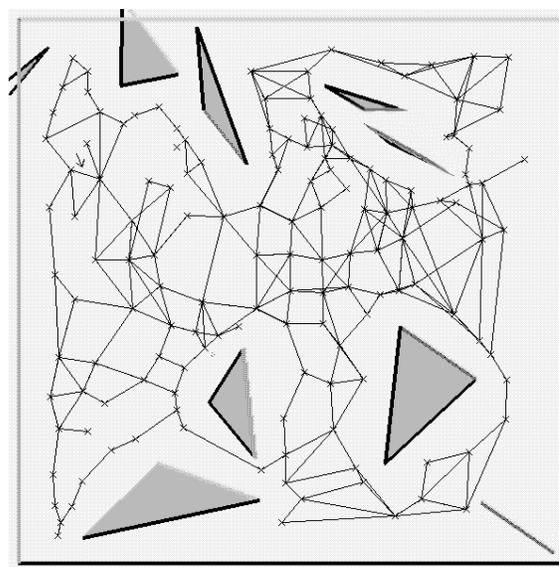


Figure 7: A simple arena, with random triangles, which was used in part of the experiments (see text). The depicted graph shows locations of snapshots and connections.

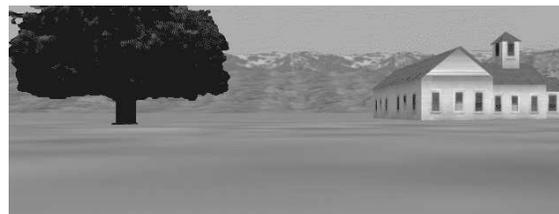


Figure 8: Virtual reality environment used in the simulations.

Mobile robot

360° sensor. In the next stage, we implemented the navigation system on a mobile robot (Fig. 9). The mechanical base consists of a six-wheeled model ground vehicle supplemented by bumpers, infrared proximity sensors and a frame for various instruments. The imaging system comprises a conical mirror mounted above a video camera which points up to the center of the cone (Fig. 10). This configuration allows for a 360° horizontal view field extending from 10° above the horizon to 30° below it. A similar imaging technique was used by Chahl and Srinivasan (1996) and Yagi, Nishizawa, & Yachida (1995).

First implementation with video transmission. In the first version, the video image was transmitted to a work station and subsequently sampled on four circles along the horizon line with a resolution of 2.5°. The circles are averaged radially to cope with errors caused by

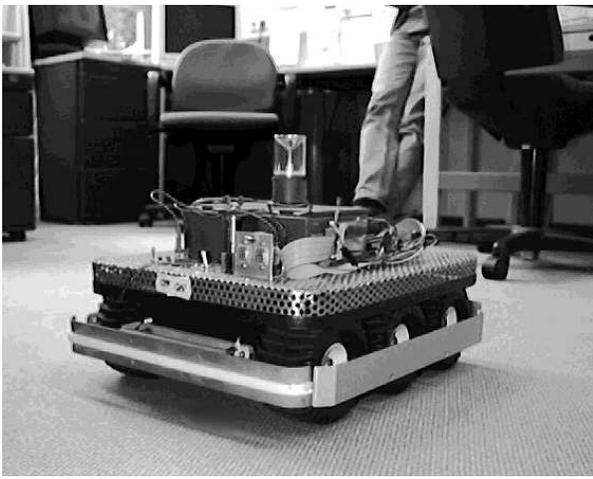


Figure 9: Mobile robot platform in the office environment where the homing experiments were performed.

misalignment of mirror and camera, tilt of the robot platform or inaccurate placing of the circles. To prevent aliasing effects and to reduce the noise level the resulting array of grey levels is spatially and temporally low pass filtered and in a final step contrast maximized. Based on this data our navigation scheme calculated movement decisions which were transmitted again via a wireless modem to an on-board Motorola 68332 controller. Besides motor control the 68332 was used to monitor and communicate the bumper and internal variables back to the work station. The slow wireless serial communication process limited the response time of the system to 200 ms.

On-board processing. To avoid the problems caused by the video transmission in indoor environments we implemented our navigation scheme in a second version



Figure 10: 360°-camera with conical mirror (scale in cm)

on the on-board controller. To provide image data directly to the controller, the video images were digitized on-board and written to memory by a Xilinx FC 4003 FPGA chip. In spite of the limited image processing capabilities of the 68332, we still were able to reach update rates up to 1.3/s because of the relatively low computational requirements of the homing algorithm. Fig. 9 shows an environment where the robot was able to home from distances of about a meter; a thorough experimental evaluation remains to be done.⁶

Miniature prototype. A third prototype has been implemented on a miniature KheperaTM robot (Fig. 11) which will allow extensive exploration runs to test the exploration strategies under real world conditions in small scale environments.



Figure 11: KheperaTM robot with camera and conical mirror.

Discussion

Our results indicate that view graph based navigation is a feasible alternative to classical robotic navigation schemes relying on metric maps. A graph representation is a useful tool for condensing the task relevant information out of the overwhelming influx of data provided by image sensors. Moreover, the integration of other information sources in a common graph representation is a straightforward procedure. Nodes may contain information about different input and internal states. Lieblich and Arbib (1982), for instance, argue that animals use a graph where nodes correspond to recognizable situations. A generalization of purely

⁶For an MPEG movie of the robot in action, check <http://www.mpik-tueb.mpg.de/people/personal/bs/sokke.html>.

topological maps are graphs where links are labelled by actions (e.g. Kuipers & Byun, 1988, Schölkopf & Mallot, 1995). This way, systems can be built which are not confined to just one type of action (in our case, this was a homing procedure). If metric information is available, graph labels can include directions to the neighbouring nodes, thus extending the allowed distance between snapshots. In an ongoing project, we are investigating an optical path integration system for this purpose. In contrast, the purely topological approach of the present study relies on the availability of a local mechanism to reach neighbouring nodes. Already this simple scheme, however, allows to solve complex navigation like path planning: once the graph has been learnt, one can generate a path to a goal by standard search algorithms, and then sequentially navigate along this path by homing. Although presented for navigation problems, similar approaches may well be feasible for other cognitive planning tasks to be performed by autonomous agents, along the lines of Tolman's (1932) *means-end-fields*, which can be conceived of as directed graphs with *means-* and *goal-objects* as nodes and *means-end-relations* as edges.

The navigation system was developed and evaluated to a great extent in virtual environments. We noticed, however, that in spite of the high degree of realism certain details of the practical implementation are easily overlooked in simulations. For example, in a simulation, the movement of the agent can be confined exactly in a plane, while views can be taken with constant orientation. The strong influence of view orientation or sensor tilt on the system's performance cannot be detected in such a simulation. Generally there is no guarantee that all relevant features of the environment have been included in the simulation which makes robotic implementations still irreplaceable.

In spite of the progress neuroscience has made in explaining behaviour, many of the underlying mechanisms remain unknown due to the complexity of the interaction of the nervous system with its environment. In contrast to the analytic way of studying behaviour and its neural substrate in increasing detail, one can approach this problem from the other end: by synthesizing behaviour using known mechanisms. We use this approach to study the most basic mechanisms of navigation behaviour. As simulations and robot implementations illustrate, these minimal strategies are sufficient to allow successful navigation under realistic environmental conditions. Since the more complex navigation mechanisms of higher animals and man developed from preexisting simpler ones, we believe that the understanding of the most basic strategies will provide the foundation for understanding and implementing high level navigation skills.

References

I. A. Bachelder and A. M. Waxman. A view-based neurocomputational system for relational map-

making and navigation in visual environments. *Robotics and Autonomous Systems*, 16:267 – 289, 1995.

R. Basri and E. Rivlin. Localization and homing using combinations of model views. *Artificial Intelligence*, 78:327 – 354, 1995.

V. Braitenberg. *Vehicles. Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA, 1984.

B. A. Cartwright and T. S. Collett. Landmark maps for honeybees. *Biological Cybernetics*, 57:85 – 93, 1987.

J. S. Chahl and M. V. Srinivasan. Visual computation of egomotion using an image interpolation technique. *Biol. Cybern.*, 74:405 – 411, 1996.

T. S. Collett. Insect navigation en route to the goal: Multiple strategies for the use of landmarks. *J. exp. Biol.*, 199:227 – 235, 1996.

C. R. Gallistel. *The Organization of Learning*. MIT Press, Cambridge, MA, 1990.

J. Hong, X. Tan, B. Pinette, R. Weiss, and E. M. Riseman. Image-based homing. In *Proc. IEEE Intl. Conf. on Robotics and Automation 1991*, pages 620 – 625, 1991.

H. G. Krapp and R. Hengstenberg. Estimation of self-motion by optic flow processing in single visual interneurons. *Nature*, 1996. in press.

B. J. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47 – 63, 1991.

I. Liebllich and M. A. Arbib. Multiple representations of space underlying behavior. *Behavioral and Brain Sciences*, 5:627 – 659, 1982.

H. Mallot, H. Bülhoff, P. Georg, B. Schölkopf, and K. Yasuhara. View-based cognitive map learning by an autonomous robot. In F. Fogelman-Soulie and P. Gallinari, editors, *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, volume II, pages 381–386. EC2, Nanterre, France, 1995.

M. J. Mataric. Navigating with a rat brain: a neurobiologically-inspired model for robot spatial representation. In J.-A. Meyer and S. W. Wilson, editors, *From Animals to Animals*. MIT Press, Cambridge, MA, 1991.

B. L. McNaughton. Neuronal mechanisms for spatial computation and information storage. In L. Nadel, L. A. Cooper, P. Culicover, and R. M. Harnish, editors, *Neural Connections, Mental Computation*. MIT Press, London, 1989.

J. O'Keefe. The hippocampal cognitive map and navigational strategies. In J. Paillard, editor, *Brain and Space*, pages 273 – 295. Oxford University Press, Oxford, 1991.

- T. Röfer. Controlling a robot with image-based homing. In *Kognitive Robotik (ZKW-Bericht)*, volume 3/95, pages 1 – 11, 1995.
- B. Schölkopf and H. A. Mallot. View-based cognitive mapping and path planning. *Adaptive Behavior*, 3:311 – 348, 1995.
- E. C. Tolman. *Purposive Behavior of Animals and Men*. Irvington, New York, 1932.
- R. Wehner, B. Michel, and P. Antonsen. Visual navigation in insects: Coupling of egocentric and geocentric information. *J. exp. Biol.*, 199:129 – 140, 1996.
- T. Wittmann. Modeling landmark navigation. In *Kognitive Robotik (ZKW-Bericht)*, volume 3/95, pages 1 – 16, 1995.
- Y. Yagi, Y. Nishizawa, and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor COPIS. *IEEE Trans. Robotics Automat.*, 11:634 – 648, 1995.