Autonomous Robots, xx, 1–11 (xx)
© xx Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Learning View Graphs for Robot Navigation

MATTHIAS O. FRANZ BERNHARD SCHÖLKOPF
HANSPETER A. MALLOT HEINRICH H. BÜLTHOFF
Max-Planck-Institut für biologische Kybernetik, Spemannstraße 38, 72076 Tübingen, Germany,
franz/bs/ham/hhb@mpik-tueb.mpg.de

Abstract. We present a purely vision-based scheme for learning a topological representation of an open environment. The system represents selected places by local views of the surrounding scene, and finds traversable paths between them. The set of recorded views and their connections are combined into a graph model of the environment. To navigate between views connected in the graph, we employ a homing strategy inspired by findings of insect ethology. In robot experiments, we demonstrate that complex visual exploration and navigation tasks can thus be performed without using metric information.

1. Introduction

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 behaviour can already be achieved using simple reactive mechanisms such as association of landmarks with movements (Wehner et al. 1996) or tracking of environmental features (Collett 1996). However, for complex navigation tasks extending beyond the current sensory horizon, some form of spatial representation is necessary. Higher vertebrates appear to construct representations — sometimes referred to as *cognitive maps* — which encode spatial relations between relevant locations in their environment (see O'Keefe & Nadel, 1978, and Gallistel, 1990, for reviews).

Under certain conditions, such maps can be acquired visually without any metric information. Humans, for instance, are able to navigate in unknown environments after presentation of sequences of connected views (e.g. O'Neill, 1991; Gillner & Mallot, 1997). In classical Artificial Intelligence research, however, robotics approaches have focused on constructing accurate global metric representations, based on a variety of mostly non-visual sensors. Besides the high computational costs, such geometric models tend to contain a large amount of irrelevant information while the cues that lead to their construction are not specifically represented. Recently, researchers have started to investigate more task-oriented representations based on topological spatial relations (e.g., Kuipers and Byun, 1991; Mataric, 1991; Bachelder and Waxman, 1995).

Many of these systems rely primarily on local sonar patterns for the identification of places which often are not distinctive enough to discriminate between similar locations. For this reason, additional metric information has to be included into the representation to facilitate the disambiguation task (e.g. compass information in Kuipers & Byun, 1991). In contrast, visual sensors can provide enough information to allow for a purely topological approach. Bachelder & Waxman (1995), for instance, have reported results on a neural control architecture based on object recognition techniques for landmark detection. In the current implementation, however, their system has to rely on artificially illuminated buildings and a pre-programmed path during exploration of the environment.

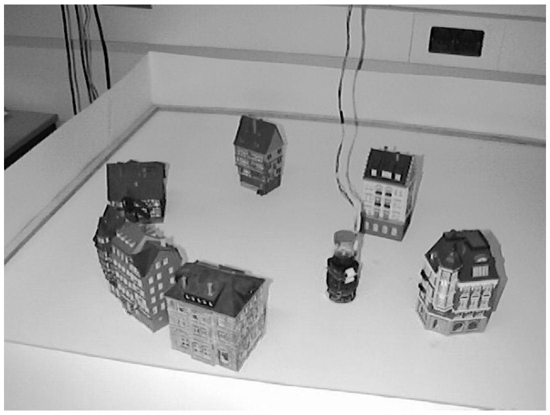

Fig. 1. 118 × 102 cm size test arena with toy houses.

In this study, we will present a system that explores its environment autonomously without making use of artificial beacons or landmarks. Based on a simple behavioural architecture, it acquires a topological representation consisting of local views and their spatial relations, termed *view graph*. For maze-like environments, Schölkopf and Mallot (1995) have shown that learning a graph of views and movement decisions is sufficient to generate various forms of navigation behaviour known from rodents. This includes finding previously visited locations, or updating spatial representations with egomotion information to maintain knowledge about one's position even when visual information is temporary unavailable. The scheme has subsequently been implemented in a mobile robot (Mallot et al., 1995). The present study undertakes to extend this approach from simple mazes to open environments. We will demonstrate its feasability and investigate some of its properties and limitations.

In the next section, we describe the experimental setup, followed by an introduction of the required basic mechanisms in Section 3, namely the procedures for homing and for selecting representative views. The integration of these procedures into a complete system will be the focus of Section 4. We conclude our study by discussing experimental results and possible extensions of the view graph approach.

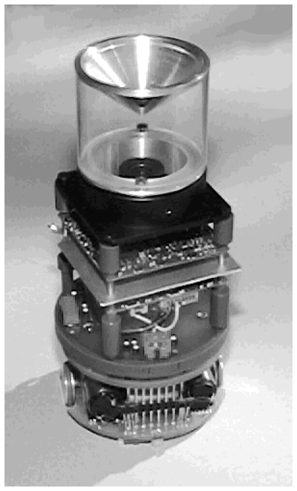

Fig. 2. Khepera^{TM} robot with camera module and custom made conical mirror, which permits sampling of the environment over 360°, in a range of ±10° about the horizon.

2. Experimental setup

Robot experiments were conducted in an arena sized 118 × 102 cm. Visual cues were provided by model houses and landmarks surrounding the arena (see Fig. 1). We used a modified Khepera miniature robot (Fig. 2) connected to an SGI Indy workstation via a serial and video transmission cable. Obstacles in a range between 0.5 cm and 2 cm were detected with 8 infrared proximity detectors.

The imaging system on the robot comprises a conical mirror mounted above a small video camera which points up to the center of the cone (Fig. 2). This configuration allows for a 360° horizontal field of view extending from 10° below to 10° above the horizon. A similar imaging technique was used by Chahl and Srinivasan (1996) and Yagi, Nishizawa, & Yachida (1995). The video image was sampled on four rings along the horizon with a resolution of 4.6° and averaged vertically to provide robustness against inaccuracies in the imaging system and tilt of the robot platform. In a subsequent processing stage, a spatiotemporal Wiener lowpass filter (e.g. Goldman, 1953) was applied to the resulting one-dimensional array. To remove changes in the illumination, the average background component was subtracted and, in a final step, the contrast of the array was enhanced via histogram equalization. The movement commands calculated from this data were transmitted back to the robot using a serial data link with a

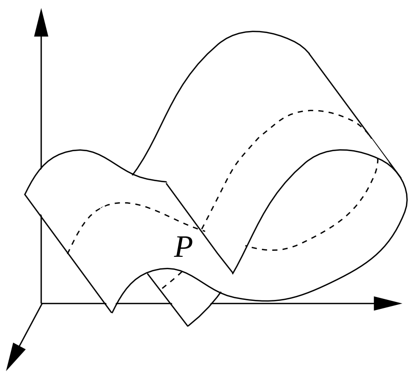

Fig. 3. Sketch of a view manifold, consisting of all views that can be seen by a continuously moving observer. The manifold is embedded in a Euclidean space whose dimensionality is the number of camera pixels. The actual structure of the manifold is much more complicated, with holes caused by obstacles, and a tubular structure due to the possibility to take snapshots at all orientations between 0° and 360°. Point P marks a singularity of the coordinate system inherited from position space: If one moves along the dotted path, the same view occurs twice at different spatial locations.

maximal transmission rate of 12 commands per second.

The Khepera's position was tracked with a camera mounted above the arena. Position and image data were recorded with a time stamp and synchronized offline. Position information was not available to the robot during the experiments.

Computer simulations were done in a two-dimensional environment consisting of triangles of random size and shading (Franz et al., 1997). Views were computed with standard ray-tracing techniques, while the control architecture was identical to the one used in the real-world experiments.

3. Basic Mechanisms for Navigating in Open Environments

3.1. Discrete Representation of Continuous Space

In view-based navigation tasks, visual information is used to guide an agent through space. The reason why this is feasible at all, is the fact that there is a continuous mapping between position space (x- and y-coordinates, possibly supplemented by gaze directions) and the space of all possible views: for each spatial position, a certain view is perceived, and this view changes continuously as the observer moves around in space.[1] Unfortunately, this mapping can be singular, as identical views can occur at different spatial locations, i.e. there is no guarantee for the existance of a global coordinate system on the manifold of all possible views (cf. Fig. 3). In principle, this problem can be dealt with using context information: In points with identical views, we can use neighbouring views to find the most likely associated position.

Complete knowledge of this manifold would be very useful to determine 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 is sufficient to store views which allow the description of relevant paths. This leads to a less detailed representation of the view manifold, namely by a graph of representative views and connections between them.

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. As open environments do not impose a structure on the view graph, we have to select a set of views which are representative for the manifold (in the following referred to as *snapshots*), and to find edges between them.

If we restrict ourselves to the use of purely topological information, i.e., we do not label the graph edges with directions, it is necessary to use a method which allows us to find connected views from a given start view. We will refer to such a method as *homing.* In the following section, we describe the two crucial components of our scheme: The procedures for homing and for taking snapshots.

3.2. Navigating between Places: Scene-based Homing

A location may be identified visually using one of two methods: First, by association with an image *of* the location (recorded while approaching or

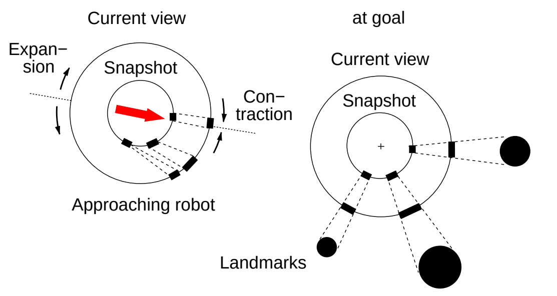

Fig. 4. A robot with omnidirectional sensor uses a snapshot taken during a previous visit to find back to the goal position. The goal direction is in the image regions with maximal contraction with respect to the snapshot (after Cartwright & Collett, 1983).

leaving it), or second, by association with an image of the scene as seen *from* the location. These two methods depend on the visual characteristics of the location and determine how such a snapshot can be used to recover its associated spatial position: If the location itself is marked by salient optical features, these can be *tracked* until the goal is reached (e.g. Collett 1996). If there are no such features, the location has to be defined by its relational properties, e.g., with respect to an array of surrounding landmarks. After a displacement, the direction back to the location can be inferred by comparing the current visual input to the snapshot: Image regions in the direction of the displacement are expanded while the image in the goal direction is contracted. Driving into the direction of maximal image contraction eventually leads to the goal position (Fig. 4).

A number of experiments have shown that invertebrates such as bees or ants are able to pinpoint a location defined by an array of nearby landmarks (see Collett 1992 for a review). Apparently, these insects search for their goal at places where the retinal image forms the best match to a memorized snapshot. Cartwright and Collett (1983) have put forward the hypothesis that bees might be able to actively extract the goal direction by a mechanism using the azimuth and size change of visible objects after a displacement.

As robots usually move in the open space between obstacles, such an approach is especially suitable for robot implementation. In order to apply the idea of Cartwright & Collett (1983) to robotic homing tasks, two basic problems have to be solved:

1. Correspondences between image points in the snapshot and in the current view must be established to detect disparities between them.
2. If a visually navigating agent has no direct access to the actual distance of the surrounding landmarks, this lack of knowledge must be compensated by some additional assumption about the distance distribution of possible landmarks in the environment.

In our approach (Franz, Schölkopf & Bülthoff, 1997), we assume that all visible landmarks have approximately the same distance from the location of the snapshot. The resulting disparity fields have a very simple structure and can be used as matched filters: From a variety of these predefined disparity fields, we determine the degree of match to the actual disparities. The best match is used to estimate the driving direction by finding the maximally contracted image region.

It can be shown mathematically that the goal can be approached with arbitrary accuracy even though the differences in the distances to the individual landmarks are neglected, and that each snapshot is surrounded by a catchment area (Franz, Schölkopf & Bülthoff, 1997). In practice, the accuracy depends mainly on the noise properties of the detector ring, since a displacement can only be detected if it generates sufficient change in the detector signal. In our experimental setup, this was usually the case at distances from the goal in the range of 1 to 3 cm, depending on the distances of the surrounding landmarks. The size of the catchment area for a single snapshot is mainly determined by the layout of the environment. In our toy house arena, maximum homing distances of 45 cm were achieved. The success rate was 95 % for homing distances smaller than 15 cm, and dropped to 50 % in the range of 20 to 25 cm.

The matching of three parameters requires minimal computational resources compared to other methods for image matching. On an SGI Indy workstation, the calculation of a home vector from 78-dimensional views took less than 40 ms which allows real time image processing at video rate. The continuous home vector computation results in smooth trajectories to the home position.

The disparities in panoramic views after a displacement have been used in several robotic systems for homing tasks. E.g., Hong et al. (1991)

identified and matched image features in images with constant orientation. They used this scheme to successfully guide a mobile robot along a corridor. In Röfer's system (1995), a Kohonen network had to learn the correspondence between snapshot and current view. Both approaches assume an approximately isotropic landmark distribution for the computation of the driving direction.

3.3. Sampling the View Manifold with Snapshots

In our robot implementation, the vertices of the view graph are identified with snapshots of the surrounding panorama, namely one-dimensional 360° records of the grey values at the horizon. While this limitation to a small subregion of the image discards potentially useful information, the omnidirectional image of the horizon has several advantages: 1. The small number of 78 pixels allows real-time processing with small computational resources. 2. Landmarks at the horizon do not leave the field of view during rotation and translation unless they become occluded. 3. Rotational and translational disparity fields sampled in a 360° field of view can easily be separated (Nelson & Aloimonos, 1988). 4. A 360° field of view makes the recognition of views relatively robust against changes affecting only parts of the panorama, e.g. moving persons. 5. Places can be encoded by a single view without the need to integrate partial views into a place representation.

Ideally, the set of snapshots taken to represent a given environment should satisfy three criteria: First, the views should be distinguishable. In purely graph-based maps, this is the only way to guarantee that specific vertices can be navigated to. This can be achieved by incorporating only distinct views into the graph. Second, a large proportion of the view manifold should be covered with a small number of vertices to keep processing requirements small. Third, the spatial distance of neighbouring views should be small enough to allow reliable navigation between them.

As we confine our system to use only visual input, the selection of the snapshots must be based on the current view and the stored snapshots. The criteria can be fulfilled by measuring the degree of similarity between views: Dissimilar views are distant on the view manifold and distinguishable by definition, and similar views often are spatially close.

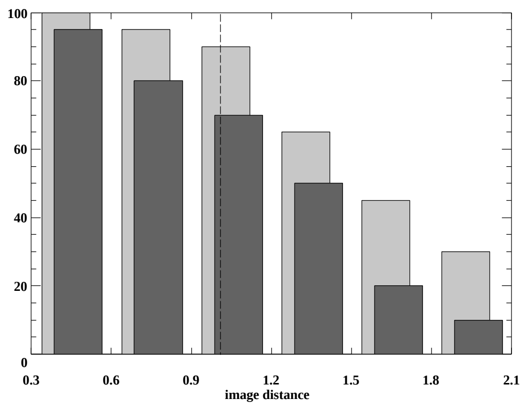

Fig. 5. Success rates (in %) for travelling between views with various image distances. In each bin, 20 pairs were tested. Light grey bars are results for pairs connected by a direct line of sight, dark bars for pairs without restrictions. The dashed line marks the threshold Θ of the classifier; image distances are measured in units of Θ.

Measuring similarity can be viewed as a pattern classification problem. We take a minimalistic approach by using the maximal pixel-wise crosscorrelation as a measure of similarity. This is equivalent to the Euclidean distance of two view vectors, after first rotating one of them such as to maximize the overlap with the other one. Whenever a threshold of the image distance to all stored snapshots is exceeded by the current view, a new snapshot is taken. The threshold is chosen to ensure that the snapshots are both distinguishable and close enough to allow safe navigation between them. Clearly, such a classifier can also be used to detect the proximity of already recorded snapshots and thus allows us to find already visited locations. We will use this classifier for both tasks in our graph learning system (see Sec. 3).

In order to find the threshold value of the image distance, we made the following experiment (Fig. 5): During a test run, the robot covered the entire free space of the arena with snapshots spaced at most 2 cm apart. From these snapshots, view pairs were selected randomly and their image distance was computed. The range of image distances was divided into 6 bins. For each bin, homing runs for 20 different view pairs were performed. A run was counted as a success if the

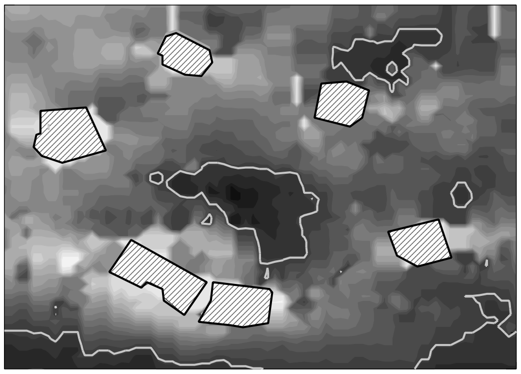

Fig. 6. Map of image distances to a snapshot at the center of the arena. Darker regions represent lower image distances. White contours enclose regions with an image distance below the threshold Θ. Hashed regions mark the shape and position of the toy houses in Fig. 1

robot reached a 2 cm circle around the home position in less than 30 s.

Since the classifier will be used for two different tasks, 20 view pairs were generated for each of the following two categories: 1. Pairs connected by a direct line of sight are relevant for the selection of snapshots, since there must be traversable space between them during exploration. 2. For edge verification (see Sec. 4.1), no restrictions on the set of possible views can be made. Figure 5 shows the success rate for both categories. We have chosen the threshold value Θ (dashed line) such that vertices connected by a direct line of sight can be reached in 90% of all cases (light grey bars). For the general case (dark bars), where the path may be blocked by obstacles, at least 70% of all vertices with image distance below Θ can still be found by the homing procedure.

As an example, we have computed the image distance map to a snapshot at the center of the arena for the view dataset mentioned above (Fig. 6). Regions with image distances below Θ are surrounded by white contours. If the robot starts at the center, the next snapshot would be taken after the white contour is crossed. This leads to a spacing of the snapshots between 5 and 15 cm, depending on the variability of the visual input. If the threshold is used to detect proximity, there are several regions with false positives. This is partly due to the low resolution and contrast of the mirror optic of the robot and partly due to the occurrence of similar views in different parts of the arena. As a consequence, the chosen

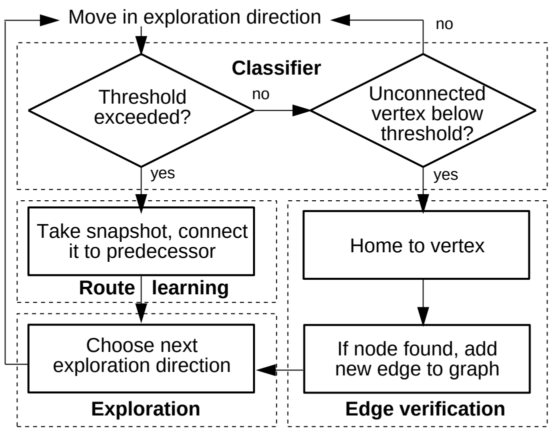
Move in exploration direction
no
Classifier
Threshold exceeded?
no
Unconnected vertex below threshold?
yes
yes
Take snapshot, connect it to predecessor
Home to vertex
Route learning
Choose next exploration direction
If node found, add new edge to graph
Exploration
Edge verification

Fig. 7. Block diagram of the graph learning algorithm.

experimental setup allows for the study of all the relevant cases discussed in Section 3.1. The number of snapshots that can be distinguished using this classifier usually falls in a range between 25 and 40, depending on the start position.

4. 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 (cf. Fig. 7):

4.1. A Minimalistic System for Learning a View Graph

Route learning. As explained before, the system uses a classifier to ensure that all recorded views are sufficiently distinct. If the image distance of the current view to the stored snapshots exceeds a threshold value, the robot takes a new snapshot and edges it to the last one. In this way, the classifier adapts the spacing between the snapshots to the rate of change in the optical input. Thus, areas which have to be covered by a denser net of snapshots, due to a rapid change of views, are also explored more thoroughly (*Kinesis*).

This route learning procedure has no way of forming new edges to previously visited views, i.e. the resulting graphs will be mere chains. By adding the following simple behaviour we can get nontrivial graphs:

Edge verification. Whenever the image distance between the current view and an unconnected vertex drops below the threshold, the robot decides to home to this vertex. If homing is successful, we include the newly learnt edge into the graph. In cases where the robot gets lost or bumps into obstacles, we start a new graph, which will typically get connected to the old one in due course. Thus, the classifier has two tasks in our system: to decide when to take snapshots and to detect candidates for overlaps between the chains of the graph.

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

Choice of exploration direction. When the robot has taken a new snapshot, or when it has homed to a vertex, a new exploration direction must be chosen. This choice primarily determines the *exploration strategy* of our system. While the classifier influences the spacing of the snapshots and the frequency of verification runs, the exploration strategy affects the overall connectivity and the spatial extent of the view graph. Clearly, a high number of stored snapshots is desirable in order to represent the environment accurately. However, if the snapshots are not sufficiently connected by edges, they are not useful for path planning. In addition, the distribution of the snapshots in space is equally important as their number. In the next section, we describe several local exploration strategies used in our system.

4.2. Local Exploration Strategies for Graph Learning

The exploration strategies used by our robot 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 qualitative guideline. In our context, knowledge gain is possible, for instance, through the recording of new edges and new snapshots. In the following, we describe several exploration strategies, which, in our case, concern primarily the choice of the next direction to explore after a snapshot has been taken, or a vertex has been reached.

Exploration direction during route learning. The simplest conceivable rule is 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. Good results can also be achieved if one uses a fixed turning angle. Using smaller angles distant areas are reached faster, whereas angles closer to π lead to a more thorough exploration of the local neighbourhood.

Exploration of the largest open angle. Our navigation scheme is designed such that all vertices of the view graph remain in the catchment areas of their respective neighbours. This property can be used to choose the next exploration direction: The system determines the home vectors to all neighbouring vertices and directs the next exploration step to the largest open angle. Alternatively, one could use information about neighbouring vertices, such as their connectivity or similarity. E.g., exploring areas where neighbouring views are connected to each other would be more likely to lead to undesired edge intersections.

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

Non-edges. The graph produced by the navigation system so far includes no information about failed actions such as obstacles encountered during exploration, or failed verification of shortcuts.

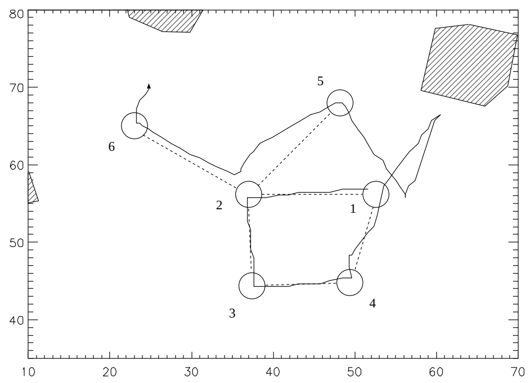

Fig. 8. Robot trajectory during an exploration run. Circles denote locations of snapshots, dotted lines recorded edges.

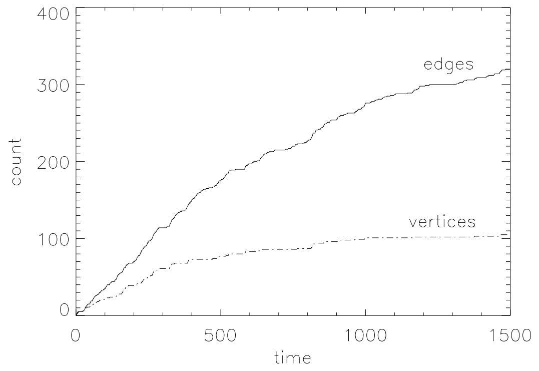

Fig. 9. Number of graph vertices and edges as a function of exploration time during a simulated exploration. The number of edges continues increasing when the number of vertices has saturated. The simulated world can be traversed in approximately 30 time units.

The exploration can be made far more effective by memorizing failed actions as "non-edges", 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 vertices, or deleting unnecessary edges. The graph structure is also influenced, to a lesser extent, by the obstacle avoidance behaviour of the robot.

Behaviour near obstacles. Distance sensors, together with low-level obstacle avoidance behaviours, are used to keep the robot away from obstacles. Typically, the visual input changes very rapidly near objects. Exploration of these areas thus requires a large number of snapshots which, in complex natural environments, would ultimately lead to a fractal graph structure near objects. To prevent the navigation system from becoming ineffective, the robot is not allowed to take new snapshots if nearby objects are detected by proximity sensors. The resulting graph structure tends to concentrate in the open space between obstacles.

Figure 8 illustrates some features of the graph learning scheme. The system starts by recording a chain of vertices 1 to 4, turning at a fixed angle of 90° after each snapshot. After leaving vertex 4, the proximity of vertex 1 is detected. The edge verification procedure establishes a new edge between vertex 1 and vertex 4 by homing to vertex 1. The next exploration direction is chosen to fall into the largest open angle, where the robot collides with an obstacle and backs up. It continues exploring until the momentary view becomes sufficiently different from all stored snapshots and starts a new graph by recording vertex 5. This time, the proximity of vertex 2 is detected, and a new edge between vertex 2 and 5 is established, connecting the two subgraphs.

The accumulation of knowledge over time can be seen from the results of a simulated exploration (Fig. 9): We first observe an increase in the number of both graph vertices and edges. 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 edges to the graph.

4.3. Examples of view graphs

Figure 10 shows two examples of view graphs G_1 and G_2 taken in the same environment with different start positions. G_1 contains 35 vertices and 50 edges, G_2 21 vertices and 32 edges. After exploration (G_1 75 min, G_2 60 min) all unconnected vertices (6 in G_1, 4 in G_2) were deleted from the graph. 89% of the edges in G_1 and 97% in G_2 could be reproduced in a subsequent homing experiment. Note, that unreproducible edges do not render the graph useless for navigation. Since the threshold of the classifier is chosen such that the

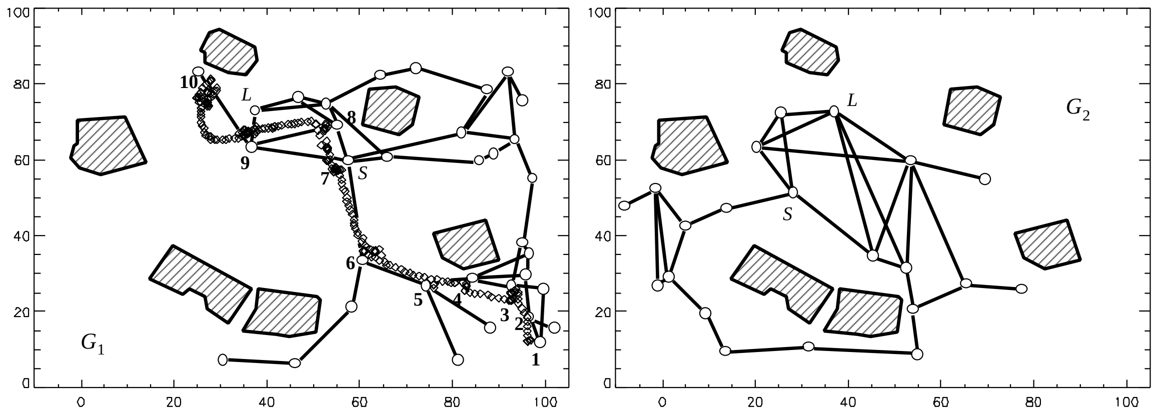

Fig. 10. Two view graphs with different start positions S. Circles denote locations of snapshots, lines recorded edges between them. The sample trajectory in G_1 started at vertex 1. Vertex L is a possible linking place between G_1 and G_2 (See discussion).

vertices remain in the catchment areas of their neighbours, the system does not lose orientation if a particular vertex cannot be found.

Both graphs cover more than half of the open space in the arena. The connectivity of the graphs reflects the topological relations of the environment. As discussed above, the system records only snapshots which are sufficiently distinguishable. This limits the number of snapshots taken in a given setup. As a consequence, systems like ours that use only topological information during exploration are necessarily confined to a subregion of the arena, where the visual information remains unambiguous.

Once the graph has been learnt, one can generate a path to a goal by search algorithms, e.g., as described by Schölkopf and Mallot (1995), and then sequentially navigate along this path by homing. The usefulness of the view graph for global navigation tasks is illustrated by the sample trajectory in G_1 (Fig. 10). The robot traverses a chain of 10 vertices, thus connecting regions which have no visual overlap.

The final analysis sheds some light on the relationship between topological (i.e. graph-based) and metrical maps. For all view pairs in the graph G_1, we computed both the graph distance (sometimes called *combinatorial distance*) and the Euclidean distance in space. It turned out that in our experimental setup, the two distance measures are strongly correlated (Fig. 11). This means that even though our system has not acquired explicit metrical information during exploration (no distances or angles were recorded), the resulting topological map does nevertheless contain some information about metric distances.

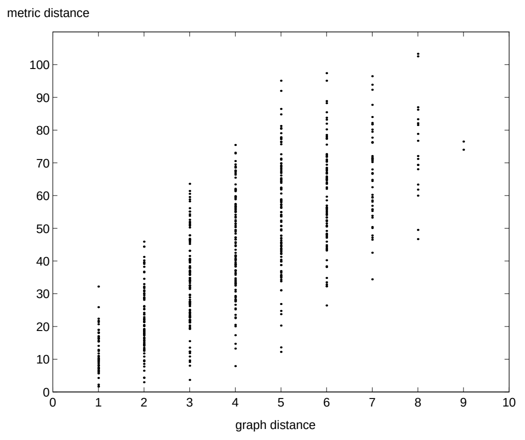

Fig. 11. Scatterplot of metric distance vs. graph distance for all connected view pairs of the graph G_1 (Fig. 10). The correlation indicates that graph distance contains information on metric distance.

5. Discussion

In this study, we presented a system which is able to acquire a graph representation of an open environment using only visual information. The purely topological approach relies on the availability of a homing mechanism to reach neighbouring vertices and a simple threshold classifier for selecting snapshots. The robot implementation demonstrates that complex exploration and navigation tasks can be performed without resorting to metric information.

Our experiments have also shown a principal limitation of systems that have no access to metric information during exploration: Only areas providing non-ambiguous information can be mapped reliably. Although the 78-dimensional views used in our system are able to encode more places than, e.g., the relatively low-dimensional signatures of ultrasonic sensor rings, the range of the system could probably be further improved by using views with higher resolution in combination with more sophisticated classifiers. In addition, views could be made more distinct by considering also their context in the graph, as proposed by Kuipers & Byun (1991) for place identification. Similarly, the neural architecture of Scölkopf & Mallot (1995) utilizes lateral weights to bias view recognition by topological context.

A simple solution to the problem of ambiguous visual input is to use distances and directions for disambiguation. Once the topological representation is learnt, no metric information is needed for navigation tasks since the vertices are uniquely defined by their context. In fact, Piaget & Inhelder (1967) proposed a similar idea in their theory of early spatial knowledge: While children do not appear to memorize explicit metric information, they seem to use it when they learn to orient themselves in their environment.

Note that disambiguation can also be performed without using metric information by including local graphs such as G_1 and G_2 in Fig. 10 in a collection of linked graphs. Common vertices between subgraphs have to be marked as *linking places* to allow transitions between them (Poucet, 1993). In our example (Fig. 10), vertex L is common both to G_1 and G_2 and could be used to switch from one subgraph to the other.

Using a purely topological representation, our system is necessarily confined to the known path segments coded in the graph. Although it is able to detect neighbouring unconnected vertices, there is no simple way to find novel paths over terrain not contained in the catchment areas of recorded views. However, our experiments have shown that our simple topological representation contains implicit metrical knowledge which might be used to accomplish tasks ususally attributed to a metrical representation. This has consequences for the interpretation of experimental results: If an animal can be shown to utilize metrical information, one cannot directly conclude that it was acquired explicitly during exploration.

Although topological approaches show certain disadvantages, it should be noted that a purely metric representation contains no explicit information on travellable paths. In cluttered environments, such a system may have problems to find its destination, because dead ends or other blockages are not represented. Thus, a combination of both approaches has the highest potential in terms of navigation performance.

Several information sources can be integrated into a common graph representation, with vertices containing information about different sensory input and internal states. Lieblich and Arbib (1982) propose that animals use a graph where vertices correspond to recognizable situations. The same idea was also used in the robot implementation of Mataric (1991) where vertices are combinations of robot motions with compass and ultrasonic sensor readings. If metric information is available, graph labels can include directions or distances to the neighbouring vertices. This allows not only for a wider spacing between snapshots but also to find shortcuts between snapshot chains over unknown terrain. A generalization of purely topological maps are graphs where edges are labelled by actions (e.g., Kuipers & Byun, 1988; Schölkopf & Mallot, 1995; Bachelder & Waxman, 1995). This way, systems can be built which do not depend on just one type of action (in our case this was a homing procedure). Although presented for navigation problems, similar graph approaches may well be feasible for other cognitive planning tasks, along the lines of Tolman (1932).

Clearly, the system we presented here is extremely simple compared to biological systems. Our intention is not to build models of animals, but to identify some of the basic building blocks that might play a role in biological navigation. This focus on understanding and synthesizing behaviour in a task-oriented way leads to parsimonious solutions with both technological and ethological implications.

Acknowledgements

The present work has profited from discussions and technical support by Philipp Georg, Susanne Huber, and Titus Neumann. We thank Fiona Newell for helpful comments on the manuscript. Financial support was provided by the Max-Planck-Gesellschaft and the Studienstiftung des deutschen Volkes.

Notes

1. If the views are recorded using sensors with overlapping Gaussian receptive fields, the view will be a smooth function of the position.

References

1. 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.
2. B. A. Cartwright and T. S. Collett. Landmark learning in bees. *J. comp. Physiol. A*, 151:521 – 543, 1983.
3. J. S. Chahl and M. V. Srinivasan. Visual computation of egomotion using an image interpolation technique. *Biol. Cybern.*, 74:405 – 411, 1996.
4. T. S. Collett. Landmark learning and guidance in insects. *Phil. Trans. R. Soc. Lond. B*, 337:295 – 303, 1992.
5. T. S. Collett. Insect navigation en route to the goal: Multiple strategies for the use of landmarks. *J. exp. Biol.*, 199:227 – 235, 1996.
6. M. O. Franz, B. Schölkopf, and H. H. Bülthoff. Homing by parameterized scene matching. In *Proc. 4th Europ. Conf. on Artificial Life*, 1997. In press.
7. M. O. Franz, B. Schölkopf, P. Georg, H. A. Mallot, and H. H. Bülthoff. Learning view graphs for robot navigation. In W. L. Johnson, editor, *Proc. 1. Intl. Conf. on Autonomous Agents*, pages 138 – 147, New York, 1997. ACM Press.
8. R. Gallistel. *The organization of learning.* MIT Press, Cambridge, MA, 1990.
9. S. Gillner and H. A. Mallot. Navigation and acquisition of spatial knowledge in a virtual maze. *J. Cognitive Neuroscience*, 1997. In press.
10. S. Goldman. *Information theory.* Dover, New York, 1953.
11. 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.
12. 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.
13. I. Lieblich and M. A. Arbib. Multiple representations of space underlying behavior. *Behavioral and Brain Sciences*, 5:627 – 659, 1982.
14. H. Mallot, H. Bülthoff, P. Georg, B. Schölkopf, and K. Yasuhara. View–based cognitive map learning by an autonomous robot. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, volume II, pages 381–386. EC2, Nanterre, France, 1995.
15. 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 Animats.* MIT Press, Cambridge, MA, 1991.
16. R. C. Nelson and J. Aloimonos. Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of you head). *Biol. Cybern.*, 1988:261 – 273, 1988.
17. J. O'Keefe and L. Nadel. *The Hippocampus as a Cognitive Map.* Clarendon Press, Oxford, 1978.
18. M. J. O'Neill. Evaluation of a conceptual model of architectural legibility. *Environment and Behavior*, 23:259 – 284, 1991.
19. J. Piaget and B. Inhelder. *The child's conception of space.* Norton, New York, 1967.
20. B. Poucet. Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms. *Psychological Rev.*, 100:163 – 182, 1993.
21. T. Röfer. Controlling a robot with image-based homing. In *Kognitive Robotik (ZKW-Bericht)*, volume 3/95, pages 1 – 11, 1995.
22. B. Schölkopf and H. A. Mallot. View–based cognitive mapping and path planning. *Adaptive Behavior*, 3:311 – 348, 1995.
23. S. Thrun. Exploration in active learning. In *Arbib, M.A. (ed.): The Handbook of brain theory and neural networks*, pages 381 – 384. MIT Press, 1995.
24. E. C. Tolman. *Purposive Behavior of Animals and Men.* Irvington, New York, 1932.
25. R. Wehner, B. Michel, and P. Antonsen. Visual navigation in insects: Coupling of egocentric and geocentric information. *J. exp. Biol.*, 199:129 – 140, 1996.
26. 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.