Autonomous Robots 5, 111–125 (1998)

Learning View Graphs for Robot Navigation

MATTHIAS O. FRANZ, BERNHARD SCHÖLKOPF,**
HANSPETER A. MALLOT AND HEINRICH H. BÜLTHOFF
Max-Planck-Institut für biologische Kybernetik, Spemannstraße 38, 72076 Tübingen, Germany
Matthias.Franz@tuebingen.mpg.de
Bernhard.Scholkopf@tuebingen.mpg.de
Hanspeter.Mallot@tuebingen.mpg.de
Heinrich.Buelthoff@tuebingen.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.

Keywords: visual navigation, topological maps, environment modeling, exploration, cognitive maps, mobile robots, omnidirectional sensor

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 and Nadel, 1978; 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). This has led to the concept of a *view graph* as a minimum representation required to explain experimentally observed navigation competences (Schölkopf and Mallot, 1995; Gillner and Mallot, 1997). A view graph is defined as a topological representation consisting of local views and their spatial relations. Depending on the task, these relations can be, e.g., movement decisions connecting the views, or mere adjacencies.

Motivated by the findings of vertebrate ethology, researchers have started to investigate topological representations for robot navigation (e.g., Kuipers and Byun, 1991; Mataric, 1991). These systems rely primarily on local sonar patterns for the identification of places, in combination with metric knowledge derived from compasses or wheel encoders. Bachelder and Waxman (1995) have reported results on a vision-based topological system which uses a neural control architecture and object recognition techniques for landmark detection. In their current implementation, however, the system has to rely on artificially illuminated landmarks and a pre-programmed path during exploration of the

*A preliminary version of this paper was presented at the First International Conference on Autonomous Agents in February 1997.
**Present address: Gesellschaft für Mathematik und Datenverarbeitung (GMD), FIRST, Rudower Chaussee 5, 12489 Berlin, Germany.

environment. 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. The scheme has subsequently been implemented in a mobile robot (Mallot et al., 1995).

It is often not clear which of the features of these robotic systems can actually be attributed to the topological representation since metrical and topological knowledge is used together. The same problem arises when interpreting biological navigation experiments since animals usually use various kinds of information depending on the task they have to perform. Therefore, it is interesting from both the robotics and the biological point of view to study systems which are restricted to the use of just one type of knowledge.

The purpose of the present study is to extend the view graph approach from the mazes of Schölkopf and Mallot (1995) to open environments. We will demonstrate that view graphs can be learned autonomously in an environment with complex visual stimuli, without making use of beacons or artificial landmarks. In doing so, we present a navigation system that uses purely topological information based on visual input. By focusing on just one type of information we want to make the contribution of topological knowledge explicit.

In the next section, we describe the experimental setup, followed by an overview of the system's architecture and an account of the required basic mechanisms, namely the procedures for selecting representative views, homing and exploration. We present experimental results in Section 4, and conclude our study by discussing possible extensions of the view graph approach.

2. Experimental Setup

Robot experiments were conducted in an arena sized 118×102 cm. Visual cues were provided by model houses (see Fig. 1). We used a modified Khepera

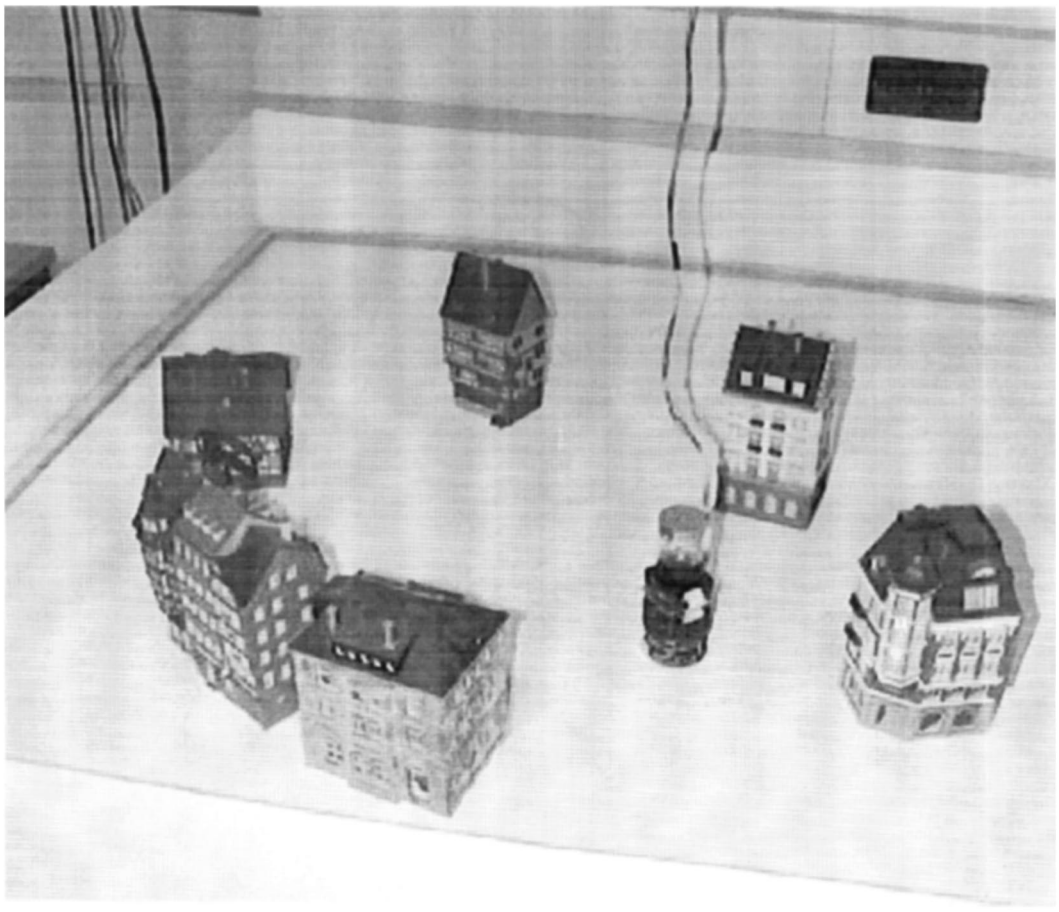


Figure 1. 118×102 cm sized test arena with toy houses and Khepera robot.

Figure 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.

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 4 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 et al. (1995).

The video image was sampled at 25 Hz 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 image data was read by an asynchronous processing thread running at 7 to 12 Hz cycle time. In each cycle, the Khepera's infrared sensors were read out using a serial data link with a maximal transmission rate of 12 Hz. The movement commands calculated from the image and infrared data were transmitted back to the robot via the same serial link.

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. Except for Figs. 10 and 11, all dataplots show results of real world experiments obtained with the described setup.

Computer simulations for Fig. 11 were done in a two-dimensional environment containing triangular objects of random size and shading (cf., Fig. 10). Views were computed with standard ray-tracing techniques. The simulated infrared sensors provided distance values corrupted by Gaussian noise. The control architecture and the interface to the simulated motors and sensors were identical to the ones used in the real-world experiments (Franz et al., 1997).

3. Learning View Graphs

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 existence 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 views from nearby spatial positions to disambiguate between them.

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

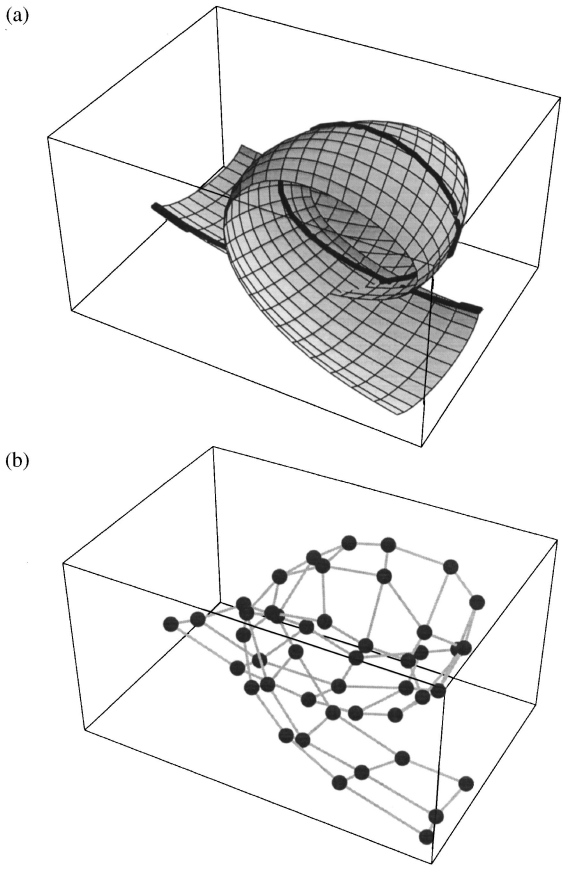


Figure 3. (a) Sketch of a view manifold, consisting of all views that can be seen by a continuously moving observer. The manifold is embedded in a Euclidean space whose dimensionality is the number of camera pixels. The actual structure of the manifold is much more complicated, with holes caused by obstacles, and a tubular structure due to the possibility to take snapshots at all orientations between 0° and 360°. The self-intersection leads to a singularity of the coordinate system inherited from position space: if one moves along the path (black line), the same view occurs twice at different spatial locations. (b) Example of a view graph representation of the view manifold.

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 consisting 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 connections between them. Since the connecting paths between the snapshots are not explicitly coded in the view graph, we have to provide a homing method which allows us to find connected views from a given start view.

In the following sections, we introduce a system that is able to solve these tasks. The vertices of the acquired view graph are panoramic views of the environment, and its edges are connections between views that can be traversed using a visual homing procedure. In contrast to our previous implementation (Schölkopf and Mallot, 1995), this homing procedure allows the system to approach a location from any direction such that the graph edges denote mere adjacency relations without any directional labelling. The resulting view graph does not contain any explicit metric information.

3.2. A Minimalistic System for Learning a View Graph

The overall architecture of the system is shown in Fig. 4. Here, we discuss the basic building blocks, the details are described in the following sections.

View Classifier. As we mentioned above, a crucial component of any graph learning scheme is the selection of vertices. The graph vertices have to be distinguishable, because otherwise the representation

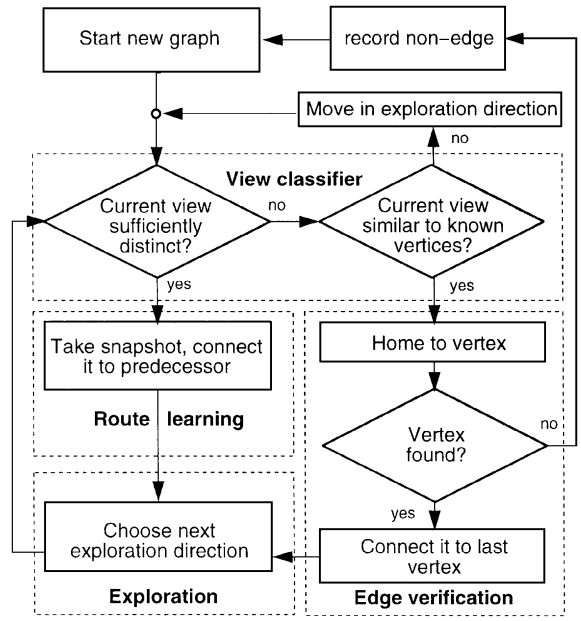


Figure 4. Block diagram of the graph learning algorithm.

could not be used for finding a specific location. Since we confine our system to use only visual information, we must guarantee that the recorded views are sufficiently distinct. This can be done by a classifier which detects whether the incoming views can be distinguished from the already stored vertices of the view graph. If this condition is fulfilled, the system takes a new snapshot and adds it to the graph. The classifier is described in Section 3.3.

Route Learning. In our system, a new snapshot is automatically connected to the previously recorded vertex of the view graph. Thus, the systems records chains of snapshots, or routes. These routes can be used to find a way back to the start position by homing to each intermediate snapshot in inverted order. We describe the homing procedure in Section 3.4. The area from which a specific snapshot can be reached by the homing procedure is called its *catchment area*. The view classifier has to make sure that every snapshot can be reached from its neighbour, i.e., all vertices of the view graph have to be in the catchment areas of their adjacent vertices.

Choice of Exploration Direction. When the system has taken a new snapshot, a new exploration direction must be chosen. This choice primarily affects the *exploration strategy* of our system, since it determines how thoroughly an area is explored and how fast the explored area grows. In the Section 3.5, we describe several local exploration strategies used in our system.

Edge Verification. The route learning procedure described above has no way of forming new edges to previously visited views, i.e., the resulting graphs will be mere chains. By adding the following behaviour we can get nontrivial graphs: During exploration, the system constantly checks whether the current view becomes similar to the already recorded snapshots. This again is a view classification task which can be solved by the same classifier as used for the selection of the snapshots (cf., Section 3.3). In a second step, the system checks whether the detected snapshot is not yet connected to the vertex from which the current exploration step started. Whenever these conditions hold, the system tries to home to the snapshot. If successful, an edge connecting the two vertices is included, and the exploration continues from the detected snapshot. In cases where the robot gets lost or bumps into obstacles, we record a "non-edge" between both vertices thus preventing the failed action from being repeated. Before starting to home, the verification procedure always checks whether a "non-edge" for this action has already been recorded. After a failed verification, we start a new graph, which will typically get connected to the old one in due course by the edge verification procedure.

If an already connected view is encountered during an exploration step the system homes to it as well (not shown in Fig. 4). 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 learned by the route learning procedure. For reasons of excessive exploration times, we did not resort to this more cautious strategy.

Arbitration and Obstacle Avoidance. As the focus of this paper is on navigation, we did not use any sophisticated obstacle avoidance systems. During exploration, the infrared sensors of the robot were continuously checked for the presence of nearby objects. If obstacles were detected at distances larger than ca., 1 cm, the robot tried to turn away without slowing down. Smaller distances were interpreted as collisions causing the robot to back up and turn away from the obstacle. Both behaviours and the graph learning system of Fig. 4 were combined into a subsumption architecture (Brooks, 1986), where the collision-induced escape behaviour had highest, the graph learning procedure lowest priority. This simple architecture proved to be sufficient to guide the robot through the environment shown in Fig. 1.

The robot is not allowed to take snapshots as long as the obstacle avoidance system is active. The resulting graph structure tends to concentrate in the open space between obstacles. This feature makes the navigation system more effective, as the visual input changes very rapidly near objects. Exploration of these areas would require a large number of snapshots which, in complex natural environments, would ultimately lead to a fractal graph structure near objects.

3.3. View Classifier

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. Omnidirectional views have the advantage that one view alone can encode a location without requiring the computationally expensive merging of several views to one place

representation. In addition, this feature provides robustness against changes in the image affecting only parts of the panorama, e.g., moving objects. Global changes in the lighting conditions are removed by the image processing (cf., Section 2) such that the view manifold in our setup is relatively stable over time.

Ideally, the set of snapshots taken to represent the view manifold 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 homing 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 above criteria can be satisfied by measuring the degree of similarity between views: Dissimilar views are distinguishable by definition while being distant on the view manifold, and similar views often are spatially close.

For measuring similarity, we take a minimalistic approach by using the maximal pixel-wise crosscorrelation Φ as a measure of similarity. This is equivalent to the dot product of two view vectors a_i, b_i, after first rotating one of them such as to maximize the overlap with the other one:

$$\Phi = \max_i \sum_j a_j b_{j-i}. \qquad (1)$$

Whenever a threshold of the image distance to all stored snapshots is exceeded by the current view, a new snapshot is taken. The threshold classifier thus adapts the spacing between the snapshots to the rate of change in the optical input. Areas which have to be covered by a denser net of snapshots, due to a rapid change of views, are also explored more thoroughly. The threshold is chosen to ensure that the snapshots are both distinguishable and close enough to allow safe navigation between them. Larger thresholds reduce the number of recordable vertices and increase the spacing between them. Smaller thresholds lead to a denser net of snapshots, but increase the danger of incorporating false edges due to the confusion of vertices. Note, that the spatial position of the snapshots is influenced to a large degree by the position of the first snapshot since all subsequent snapshots are selected according to the internal distinctiveness criterion. For varying start positions, the recorded view graph will always look different.

Clearly, a threshold classifier can also be used to detect whether the current view becomes similar to one of the already recorded snapshots. If the image distance to a snapshot falls below the threshold, the robot starts its edge verification procedure (cf., Section 3.2) and tries to home to the snapshot. In our system, we use the same classifier for both tasks. A suitable threshold was determined experimentally (cf., Section 4.1).

3.4. Navigating between Places: View-Based Homing

In order to travel between the vertices of the view graph, we need a visual homing method. Since the location of a vertex is only encoded in the recorded view, we have to deduce the driving direction from a comparison of the current view to the goal view. After the robot has moved away from the goal, the images of the surrounding landmarks in the current view are displaced from their former image positions in the goal view. If the robot moves so as to reduce these displacements it will finally find back to the goal, where current view and snapshot match. The displacement field has a focus of contraction in the goal direction (cf., Fig. 5). Driving into the direction of this focus most quickly reduces the image displacements.

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 homing mechanism as described above.

In order to apply the idea of Cartwright and 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 displacements 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.

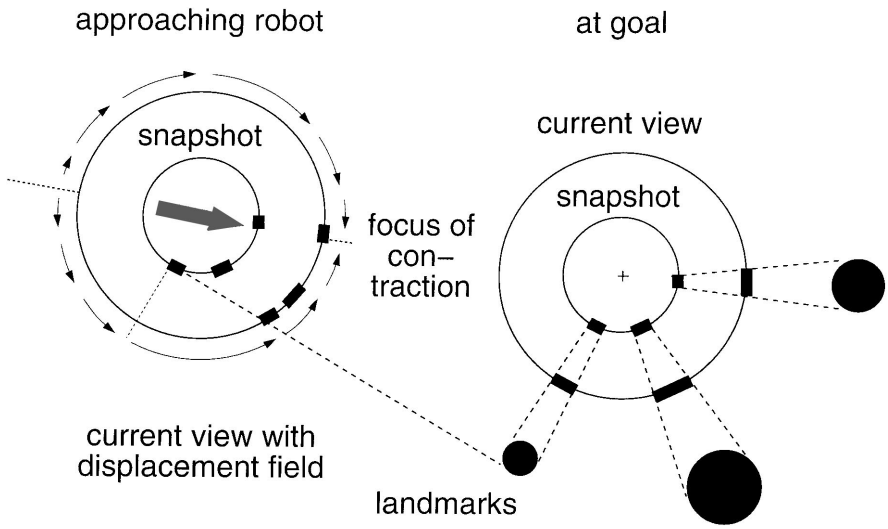


Figure 5. A robot with omnidirectional sensor uses a snapshot taken during a previous visit to find back to the goal position. After moving away from the goal, the images of the surrounding landmarks are displaced to new image positions, as denoted by the arrows at the current view. If the robot moves such that it reduces the displacements it will finally find back to the goal (after Cartwright and Collett, 1983).

In our approach, we assume that all visible landmarks have approximately the same distance from the location of the snapshot. The resulting displacement fields $\delta(\theta)$ have a very simple structure (Franz et al., 1997)

$$\delta(\theta) = \arctan\left(\frac{\nu \sin(\theta - \alpha)}{1 - \nu \cos(\theta - \alpha)}\right) - \psi, \qquad (2)$$

where θ denotes the image position of a landmark in the snapshot, α the direction in which the robot has moved away from the goal, ψ the change in the sensor orientation, and ν the ratio between the distance from the goal and the average landmark distance. Since these displacement fields depend only on three free parameters ψ, α and ν, they can be used as matched filters: From the parameters of the best match to the actual displacement field, we compute the driving direction given by $\alpha + \pi$. These steps are repeated until the current view becomes identical to the snapshot.

```
REPEAT {
  FOR all values of ψ, α, ν DO {
    compute displacement field from Eq.(2)
    distort snapshot with displ.field
    compute image distance to current view
  }
  select parameter set with min.distance
  drive in direction α + π.
}
UNTIL image distance to snapshot is 0
```

The matching of three parameters requires relatively small 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 results in smooth trajectories to the home position. It can be shown mathematically (Franz et al., 1997) 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 non-empty catchment area. In practice, the accuracy depends mainly on the quantization and sensor noise 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 displacements in panoramic views after observer movements have been used in several robotic systems for homing tasks. For example, 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 was used to learn the correspondence between snapshot and current view.

3.5. 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 concern primarily the choice of the next direction to explore after a snapshot has been taken, or after an existing vertex has been reached (cf., Section 3.2).

Exploration Direction during Route Learning. The simplest conceivable rule is to choose a random direction and then to go straight until the next snapshot is taken. 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. For the experiments presented here, we used a fixed turning angle of 90° to the left.

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, if a vertex has already more than one edge: The system determines the directions to all neighbouring vertices using the homing procedure, and directs the next exploration step to the largest open angle between them (cf., Fig. 6). Alternatively, one could use information about neighbouring vertices, such as their connectivity or similarity. For example, exploring areas where neighbouring views are connected to each other would be more likely to lead to possibly undesired edge intersections.

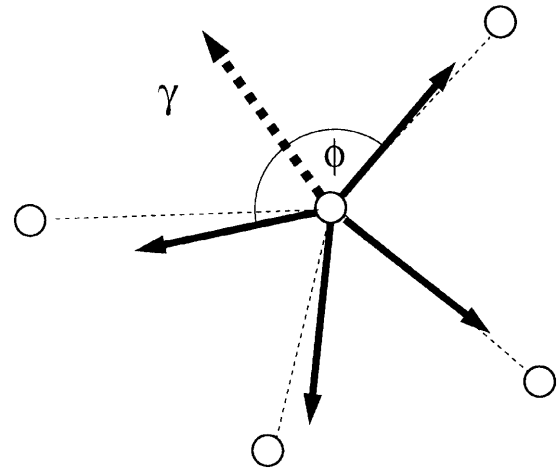


Figure 6. The robot estimates the directions to all connected vertices (black arrows). The largest open angle is denoted by Φ. The next exploration step will be directed along γ into the middle of Φ.

Limiting the Connectivity of Vertices. The exploration can be made more effective by limiting the number of edges a vertex can have. If a vertex reaches the maximum number of edges the robot moves on to less connected neighbours instead of starting a new exploration step. Similarly, the robot determines the directions to all neighbours as described above. If the largest open angle between the directions is smaller than a preset value, the system moves on to other vertices. Using this strategy, exploration tends to spread out to less explored vertices and areas.

Together, these strategies form the routine for the choice of the next exploration direction in Fig. 4:

```
Choose next exploration direction:
n := count number of edges
FOR all connected views DO
  compute directions to connected views
φ := largest open angle
γ := vector pointing into the middle of φ
IF n > max edges OR φ < min angle THEN
  move to least connected neighbour
IF n < 2 THEN
  exploration direction := fixed angle
ELSE
  exploration direction := γ
```

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.

4. Results

4.1. View Classifier Threshold

In order to find the threshold of the image distance for the view classifier, we made the following experiment (Fig. 7): 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 robot reached a 1 cm radius around the home position in less than 30 s.

Since the classifier is 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 Section 3.2), no restrictions on the set of possible views can be made. Figure 7 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. 8). 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. The number of snapshots that can be distinguished using this classifier usually falls in a range between 20 and 40, depending on the start position. This is due to the low resolution and contrast of the mirror optic of the robot and the occurrence of similar views in different parts of the arena.

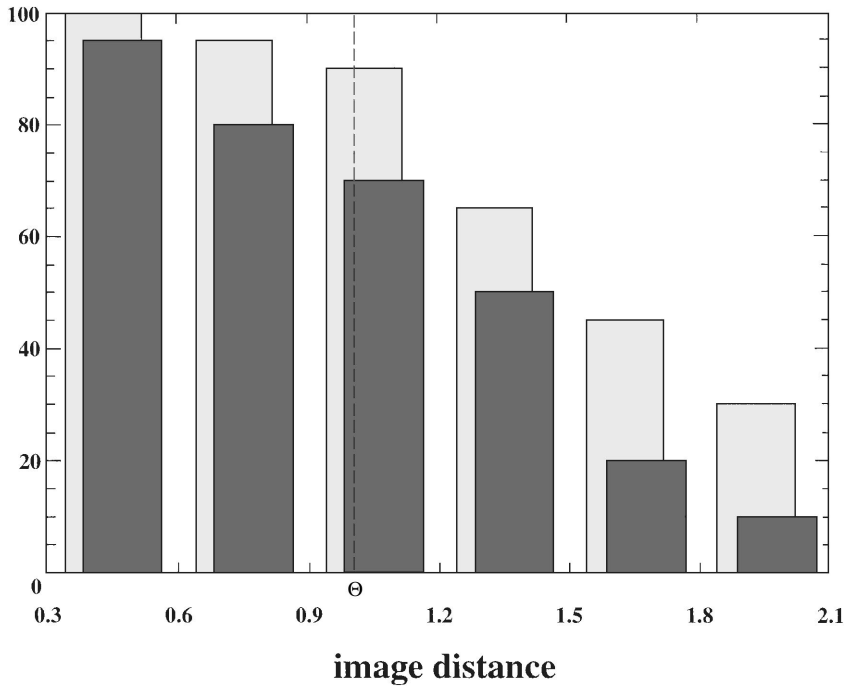


Figure 7. Success rates (in %) for travelling between views with various image distances. In each bin, 20 pairs were tested. Light grey bars are results for pairs connected by a direct line of sight, dark bars for general pairs. The dashed line marks the threshold Θ of the classifier; image distances are measured in multiples of Θ.

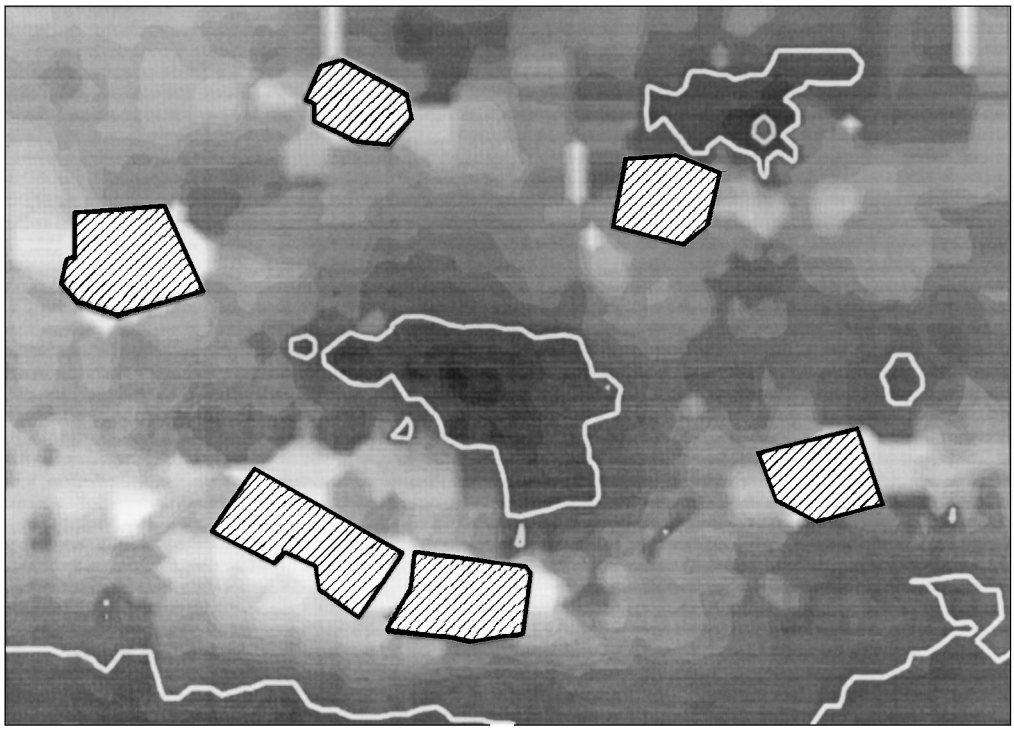


Figure 8. Map of image distances to a snapshot at the center of the arena. Darker regions represent lower image distances. White contours enclose regions with an image distance below the threshold Θ. Hashed regions mark the shape and position of the toy houses in Fig. 1.

If the threshold is used to detect similarity, there are several regions with false positives. In these regions, the edge verification procedure would try to find new edges to the central view, but fail to find it since the view never becomes similar enough to the snapshot. Although our setup allows only to record 20–40 snapshots, the visual input is sufficiently rich to prevent false edges due to identical views in different locations. Clearly, in some environments, e.g., in corridors, identical views may be more likely to occur, and thus would require additional information to disambiguate the locations.

4.2. Graph Learning

Figure 9 shows some features of the graph learning scheme. The trajectories were recorded using the tracking system described above. The robot 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 current view becomes similar to vertex 1. 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.

In order to illustrate the accumulation of knowledge over time, we used a simulated two-dimensional environment (cf., Fig. 10) as described in Section 2. This environment provided sufficiently rich visual input to record more than 100 snapshots in the view graph and allowed for long exploration times. The simulated world can be traversed in 30 time units, thus in the overall time period depicted in Fig. 11, the arena was crossed approximately 50 times. 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.

Figure 12 shows two examples of view graphs G_1 and G_2 recorded by the Khepera robot in the toy house

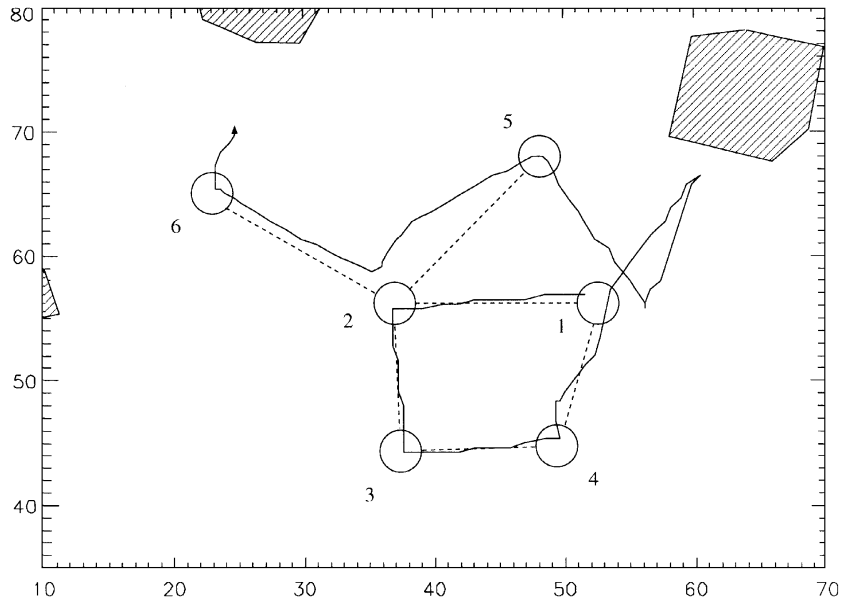


Figure 9. Robot trajectory recorded by the tracking system during an exploration run. Circles denote locations of snapshots, dotted lines recorded edges.

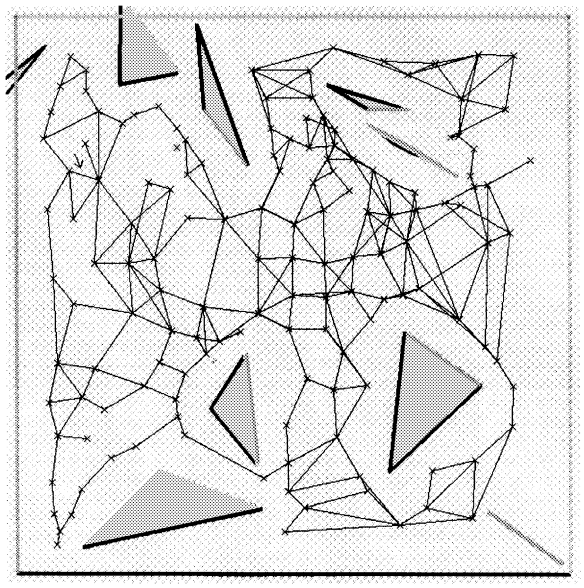


Figure 10. Simulated two-dimensional world with triangular objects of random size and shading. The depicted graph shows the location of snapshots and recorded edges between them.

environment with different start positions. G_1 contains 35 vertices and 46 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 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 prevents the robot from taking snapshots in the regions outside the areas covered by the graph because there the view is too similar to the already recorded snapshots. Nevertheless, the visual input provided by our setup is sufficiently rich that no exactly identical views occured in the arena such that no false links were recorded.

4.3. Examples of View Graphs

Once the graph has been learned, 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 use of the view graph for global navigation tasks is illustrated by the sample trajectory in G_1 (Fig. 12). The robot traverses

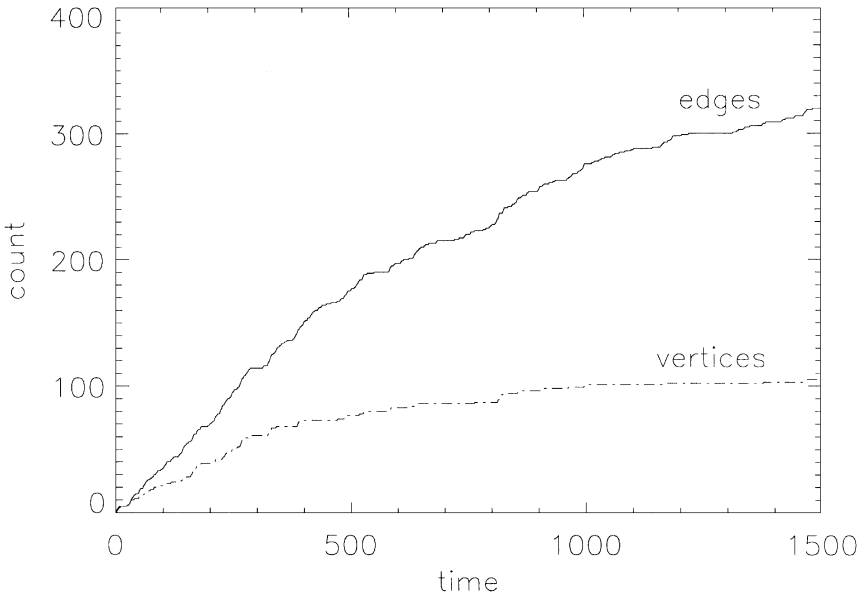


Figure 11. Number of graph vertices and edges as a function of exploration time during a simulated exploration. The number of edges continues increasing when the number of vertices has saturated.

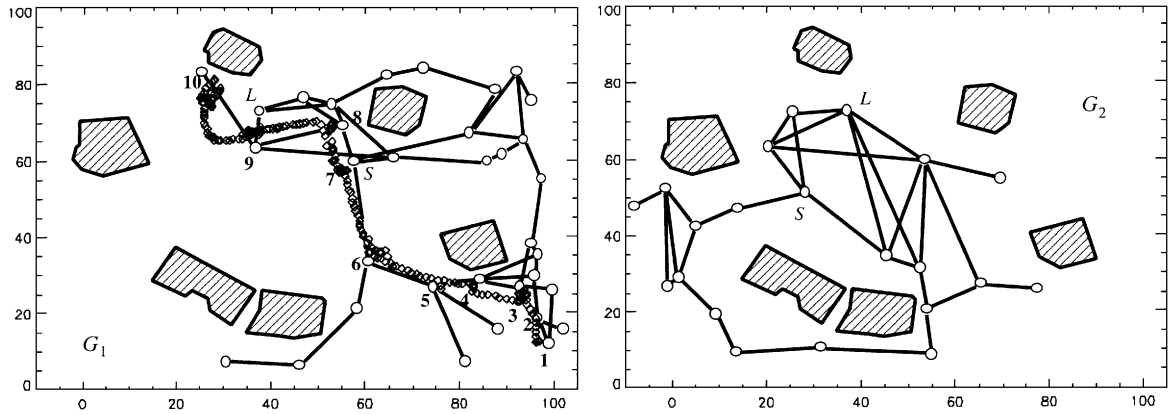


Figure 12. Two view graphs recorded by the Khepera robot with different start positions S. Circles denote locations of snapshots, lines recorded edges between them. The sample trajectory (diamonds) in G_1 started at vertex 1. Vertex L is a possible linking place between G_1 and G_2 (see discussion).

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 (or *combinatorial distance*) and the Euclidean distance in space. It turned out that in our experimental setup, the two distance measures are strongly correlated (Fig. 13). 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. This is due to the properties of the graph learning system. If two vertices are spatially far apart, any possible edge connecting them would probably intersect other edges, which our system tries to avoid (cf., Section 3.2). If two vertices are close, the edge verification procedure is likely to connect them in the graph.

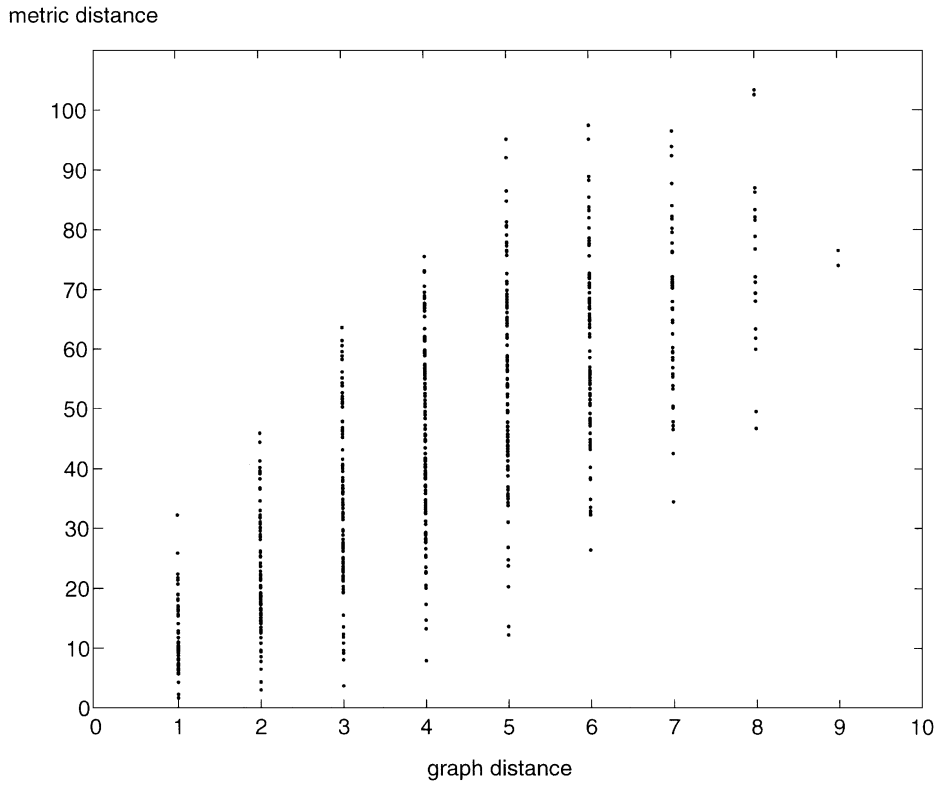


Figure 13. Scatterplot of metric distance vs. graph distance for all connected view pairs of the graph G_1 (Fig. 12). The correlation indicates that graph distance contains information on metric distance.

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 and contrast 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 and Byun (1991) for place identification. Similarly, the neural architecture of Schölkopf and 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 during learning. Once the topological representation is learned, no metric information is needed for navigation tasks since the vertices are uniquely defined by their context. Piaget and 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. 12 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. 12), 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 implications 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.

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 and Byun, 1991; Schölkopf and Mallot, 1995; Bachelder and 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, as, e.g., in the means-end-fields 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.

Acknowledgments

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

Note

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

References

Bachelder, I.A. and Waxman, A.M. 1995. A view-based neurocomputational system for relational map-making and navigation in visual environments. *Robotics and Autonomous Systems*, 16:267–289.

Brooks, R.A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1).

Cartwright, B.A. and Collett, T.S. 1983. Landmark learning in bees. *J. Comp. Physiol. A*, 151:521–543.

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

Collett, T.S. 1992. Landmark learning and guidance in insects. *Phil. Trans. R. Soc. Lond. B*, 337:295–303.

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

Franz, M.O., Schölkopf, B., and Bülthoff, H.H. 1997. Homing by parameterized scene matching. In *Fourth Europ. Conf. on Artificial Life*, P. Husbands and I. Harvey (Eds.), MIT Press: Cambridge, MA, pp. 236–245.

Franz, M.O., Schölkopf, B., Georg, P., Mallot, H.A., and Bülthoff, H.H. 1997. Learning view graphs for robot navigation. In *Proc. 1. Intl. Conf. on Autonomous Agents*, W.L. Johnson (Ed.), ACM Press: New York, pp. 138–147.

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

Gillner, S. and Mallot, H.A. 1997. Navigation and acquisition of spatial knowledge in a virtual maze. *J. Cognitive Neuroscience*. In press.

Goldman, S. 1953. *Information Theory*, Dover: New York.

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

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

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

Mallot, H., Bülthoff, H., Georg, P., Schölkopf, B., and Yasuhara, K. 1995. View-based cognitive map learning by an autonomous robot. In *Proc. ICANN'95—Int. Conf. on Artificial Neural Networks*, F. Fogelman-Soulié and P. Gallinari (Eds.) EC2, Nanterre, France, Vol. II, pp. 381–386.

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

O'Keefe, J. and Nadel, L. 1978. *The Hippocampus as a Cognitive Map*, Clarendon Press: Oxford.

O'Neill, M.J. 1991. Evaluation of a conceptual model of architectural legibility. *Environment and Behavior*, 23:259–284.

Piaget, J. and Inhelder, B. 1967. *The Child's Conception of Space*, Norton: New York.

Poucet, B. 1993. Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms. *Psychological Rev.*, 100:163–182.

Röfer, T. 1995. Controlling a robot with image-based homing. Center for Cognitive Sciences, Bremen, Report 3/95.

Schölkopf, B. and Mallot, H.A. 1995. View-based cognitive mapping and path planning. *Adaptive Behavior*, 3:311–348.

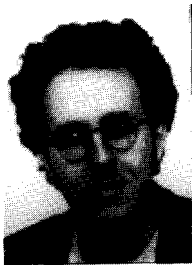
Thrun, S. 1995. Exploration in active learning. In *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib (Ed.), MIT Press, pp. 381–384.

Tolman, E.C. 1932. *Purposive Behavior of Animals and Men*, Irvington: New York.

Wehner, R., Michel, B., and Antonsen, P. 1996. Visual navigation in insects: Coupling of egocentric and geocentric information. *J. Exp. Biol.*, 199:129–140.

Yagi, Y., Nishizawa, Y., and Yachida, M. 1995. Map-based navigation for a mobile robot with omnidirectional image sensor COPIS. *IEEE Trans. Robotics Automat.*, 11:634–648.

Matthias O. Franz graduated with a M.Sc. in Atmospheric Sciences from SUNY at Stony Brook, NY, in 1994, and with a Diplom in Physics from the Eberhard-Karls-Universität, Tübingen, Germany, in 1995. He is currently a Ph.D. student at the Max-Planck-Institut für biologische Kybernetik. His main research activities are in autonomous navigation, egomotion extraction from optic flow and natural image statistics.



Bernhard Schölkopf received an M.Sc. degree in Mathematics from the University of London in 1992. Two years later, he received the Diplom in Physics from the Eberhard-Karls-Universität, Tübingen, Germany. He recently finished his Ph.D. in Computer Science at the Technische Universität Berlin, with a thesis supervised by Heinrich Bülthoff and Vladimir Vapnik (AT&T Research). He has worked on machine pattern recognition for AT&T and Bell Laboratories. His scientific interests include machine learning and perception.

Hanspeter A. Mallot studied Biology and Mathematics at the University of Mainz where he also received his doctoral degree in 1986. He was a postdoctoral fellow at the Massachusetts Institute of Technology in 1986/87 and held research positions at Mainz University and the Ruhr-Universität-Bochum. In 1993, he joined the Max-Planck-Institut für biologische Kybernetik. In 1996/97, he was a fellow at the Institute of Advanced Study, Berlin. His research interests include the perception of shape and space in humans and machines, as well as neural network models of the cerebral cortex.

Heinrich H. Bülthoff is Director at the Max-Planck-Institute for Biological Cybernetics in Tübingen, Germany and Honorar Professor at the Eberhard-Karls-Universität Tübingen. He studied Biology in Tübingen and Berlin and received his doctoral degree in 1980. He spent 3 years as a visiting scientist at the Massachusetts Institute of Technology and joined in 1988 the faculty of the Department of Cognitive and Linguistic Sciences at Brown University. In 1993, he was elected as Scientific Member of the Max-Planck Society and currently directs at the Max-Planck-Institute in Tübingen a group of about 30 biologists, computer scientists, mathematicians, physicists and psychologists working on psychophysical and computational aspects of higher level visual processes. Dr. Bülthoff has published more than 50 articles in scholarly journals in the areas of Object and Face Recognition, Integration of Visual Modules, Sensori-Motor Integration, Autonomous Navigation and Artificial-Life. He has active collaborations with researchers at Brown University, Massachusetts Institute of Technology, NEC Research Institute, Oxford University, Tübingen University, Tel Aviv University, University of Minnesota, University of Western Ontario, University of Texas and the Weizmann Institute of Science.