A robot system for biomimetic navigation - from snapshots to metric embeddings of view graphs

Matthias O. Franz[1] **Wolfgang Stürzl**[2]
Wolfgang Hübner[2] **Hanspeter A. Mallot**[2]
[1]MPI für biologische Kybernetik, Spemannstraße 38, D-72076 Tübingen, Germany
[2]Universität Tübingen, Kognitive Neurowissenschaften, D-72076 Tübingen, Germany
e-mail: Matthias.Franz@tuebingen.mpg.de;
wolfgang.stuerzl; wolfgang.huebner; hanspeter.mallot@uni-tuebingen.de

Abstract

Complex navigation behaviour (way-finding) involves recognizing several places and encoding a spatial relationship between them. Way-finding skills can be classified into a hierarchy according to the complexity of the tasks that can be performed [5]. The most basic form of way-finding is *route navigation*, followed by *topological navigation* where several routes are integrated into a graph-like representation. The highest level, *survey navigation*, is reached when this graph can be embedded into a common reference frame.

In this paper, we present the building blocks for a biomimetic robot navigation system that encompasses all levels of this hierarchy. As local navigation method, we use scene-based homing. In this scheme, a goal location is characterized either by a panoramic snapshot of the light intensities as seen from the place, or by a record of the distances to the surrounding objects. The goal is found by moving in the direction that minimizes the discrepancy between the recorded intensities or distances and the current sensory input. For learning routes, the robot selects distinct views during exploration that are close enough to be reached by snapshot-based homing. When it encounters already visited places during route learning, it connects the routes and thus forms a topological representation of its environment termed view graph. The final stage, survey navigation, is achieved by a graph embedding procedure which complements the topologic information of the view graph with odometric position estimates. Calculation of the graph embedding is done with a modified multidimensional scaling algorithm which makes use of distances and angles between nodes.

1 Introduction

Navigation behaviour in animals can be conveniently categorized into a hierarchy [5]. The layers of the hierarchy correspond to increasing complexity levels of the navigation tasks that can be performed by the animal. A major distinction in this hierarchy is that of local navigation and way-finding: *Local navigation* behaviours such as aiming, guidance, path integration etc. are used to find a single goal by using only currently available sensory information, without the need of representing any objects or places outside the current sensory horizon [17]. It requires the recognition of only one location, namely the goal. *Way-finding* involves the recognition of several places, and the representation of relations between places which may be outside the current range of perception [14]. It relies on local navigation skills to move from one place to another, but it allows the animal to find places that could not be found by local navigation alone. Way-finding behaviours can be categorized into three subsequent levels: recognition-triggered response, topological navigation and survey navigation.

Recognition-triggered responses connect two locations by a local navigation method, i.e., an association between a sensory pattern defining the start location and a motor action. In this context, a location is defined as a certain sensory situation in which the same local navigation method is selected. The recognition of the starting location triggers the activation of a local navigation method leading to the goal. There is no planning of a sequence of subsequent movements, only the selection of the very next action. Thus, the animal responds in an inflexible manner to the current situation.

Several recognition-triggered responses can be concatenated to *routes*. Routes are sequences of recognition-triggered responses, in which the goal of one step is the start of the next. The local navigation method can be different in each step according to the local environment. A route may connect locations that cannot be reached by local navigation alone. Still there is no planning in-

volved, as knowledge is limited to the next action to perform. If one route segment is blocked, e.g. by an obstacle, the animal has to resort to a search strategy until it reaches a known place again.

Topological navigation. An animal using recognition-triggered responses is confined to using always the same sequences of locations. Routes are generated independently of each other and each goal needs its own route. Navigation is more adaptive if the spatial representation is goal-independent, i.e. if the same representation can be used for multiple goals. To this end, the animal must have the basic competence of detecting whether two routes pass through the same place. Two possibly different sensory configurations associated with the different routes leading through the same place have to be merged by *route integration*. A collection of integrated routes thus becomes a topological representation of the environment. This can be expressed mathematically as a graph, where vertices represent places and edges represent a local navigation method connecting two vertices.

Any vertex can become the start or the goal of a route, so that, in the case of obstacles, alternative intersecting routes may be found. The fact that alternative routes may lead to one goal requires *planning* abilities which generate routes from the graph. Planning together with route integration are the capabilities required for *topological navigation*. The resulting routes are concatenations of sub-sequences from already visited routes. As a consequence, an animal relying on topological navigation cannot generate novel routes over unvisited terrain.

Survey navigation. Whereas for topological navigation different routes have to be integrated locally, *survey navigation* requires the *embedding* of all known places and of their spatial relations into a common frame of reference. In this process, the spatial representation must be manipulated and accessible as a whole, so that the spatial relation between any two of the represented places can be inferred. In contrast, topological navigation needs only the spatial relations between connected places. An animal using survey navigation is able to find novel paths over unknown terrain, since the embedding of the current location into the common frame of reference allows the animal to infer its spatial relation to the known places. Examples include finding of shortcuts in unknown terrain between unconnected routes, or detours over unknown terrain around obstacles.

Biomimetic navigation. Generally, each level of the navigation hierarchy requires new skills on top of the lower level skills. This could also indicate the direction taken during evolution, since new behavioural capabilities are usually built on pre-existing simpler mechanisms. A distinctive feature of a biomimetic robot way-finding system is, therefore, the use of a hierarchy of competences and their underlying mechanisms that should reflect an "evolutionary scaling" as discussed in [11]. The

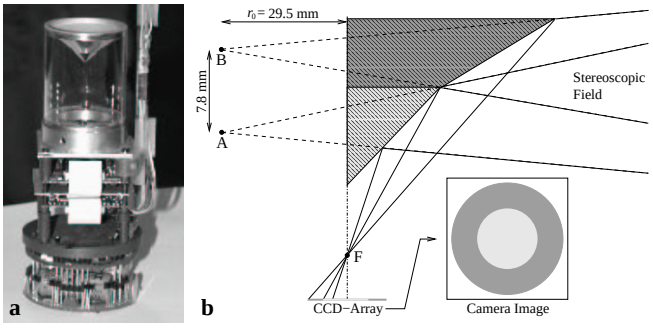

Figure 1: **a:** Khepera with panoramic stereo camera on top (diameter ≈ 5 cm, height ≈ 13 cm). **b:** Schematic diagram of the bipartite mirror for an axial plane (not to scale). The imaging can be considered as "looking" through two vertically separated points (A, B) which are mirror images of the nodal point of the camera (F). The inset shows the resulting panoramic stereo image: The inner filled circle (light grey) depicts the part imaged through the lower cone; the outer part (dark grey) is imaged through the upper cone.

approach of technical robotics to navigation is most reminiscent to survey navigation since spatial knowledge is represented in a common global map. This contrasts with the above considerations in which survey navigation is the very last stage of the evolutionary development. Biomimetic approaches are therefore constructed in a bottom-up manner: Higher navigation abilities are used on top of simple, but reliable mechanisms. Sometimes these simpler mechanisms turn out to be sufficient for a given task, so that the higher levels need not to be implemented.

Several biomimetic navigation systems for recognition-triggered responses and topological navigation exist in the literature (see [5], for an overview). The final step to survey navigation still awaits its robotic implementation. In the following, we present the building blocks for such a robotic survey navigation system that encompasses all three levels of way-finding. Route and topological navigation are already implemented on a mobile robot, survey navigation works so far only in simulations. All experiments were done using a Khepera miniature robot in a toy house arena of approximately 1m^2 size. As local navigation method, we use a scene-based homing procedure (Sect. 2). The implementation and algorithms for the subsequent levels of recognition-triggered response, topological and survey navigation are described in Sects. 3, 4 and 5. We conclude in Sect. 6 by discussing the results obtained so far.

2 Scene-based homing

Bees or ants are able to use visual guidance (scene-based homing) as they find a location which is only defined by

its spatial relationship an array of locally visible landmarks (for review, see [3]). The experimental evidence suggests that these insects store a relatively unprocessed snapshot of the surrounding panorama as seen from the goal. Cartwright & Collett [1] developed a computational model that allowed to find the goal by matching the snapshot with the current view. Computer simulations showed that the model could indeed account for the observed search behaviour of honeybees.

This simple form of visual guidance has inspired several robot implementations since no complex scene representations have to be handled to find an inconspicuous goal (overview in [5]). As robots usually move in the open space between obstacles, scene-based homing is especially suitable for robot navigation. Our own approach [7] used unprocessed panoramic images of the light intensities seen at the horizon. Under constant lighting conditions, our robot showed robust homing performance. However, when lighting conditions changed completely between taking the snapshot and homing (as, e.g., from sunlight to artificial illumination), the performance broke down [16]. This suggested a natural extension of the original scheme: Instead of using unprocessed grey values, one could use a "snapshot" of the distances to the surrounding objects at the goal position since the distance distribution in a scene is invariant under illumination changes. There is also strong evidence that rodents, see e.g. [2], [4], and also humans, e.g. [8], use memorized geometric cues to return to already visited places.

The resulting homing algorithm used inverse distances (disparities) to the surrounding objects as snapshots for computational reasons (cf. Sect. 2.1). It showed robust performance with respect to changes in the lighting conditions. However, the area around the goal from which the goal can be found, i.e., the catchment area of the goal, was slightly smaller than in the original, grey-value based scheme [16]. Homing accuracy depends mainly on the noise properties of the imaging device, since a displacement can only be detected if it generates sufficient change in the image. 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. In the remainder of this section, we describe the disparity-based homing scheme in detail. Both homing schemes, view-based and disparity-based, are used as local navigation method in the way-finding system described in the subsequent sections.

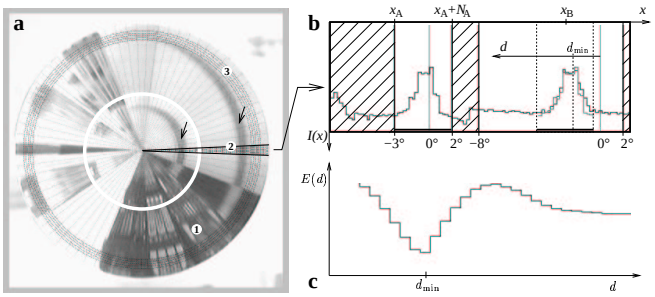

Figure 2: **a:** Raw stereo image. Images of the toy house environment can be seen in the lower right part (1). In the marked sector element (2), a horizontal line on the arena wall is imaged twice (arrows). **b:** Grey values corresponding to the sector element in **a**. Linear search for maximum correlation (error function plotted in **c**) between the inner and outer part yields the disparity. The hatched parts are excluded because of low horizontal resolution in the image center (left) and because of imaging distortions at the transition area of the two different slopes of the mirror (middle).

2.1 Disparity signatures of places

In order to acquire geometric information of the robot's current place we have built a panoramic stereo sensor. Mounted on top of a Khepera miniature robot, a CCD-camera is directed vertically towards a bipartite conic mirror (see Fig. 1 a). It consists of two conical parts with slightly different slopes yielding an effective vertical stereo base line of ≈ 8 mm (Fig. 1 b).

As depicted in Fig. 2 a, raw stereo images, taken by the panoramic stereo sensor, are divided into $N = 72$ sectors (representing a 5° range horizontally). Each sector is subdivided into radial elements resulting in an array of 100 grey-scale pixels $I(x)$, $x = 0, 1, \ldots, 99$ (Fig. 2 b).

We have implemented a simple correlation based stereo algorithm to estimate the mean shift d (disparity) of the two image parts by minimizing the matching error (see Fig. 2 b,c),

$$d_{\min} := \arg\min_d E_m(d) \quad (1)$$

$$E_m(d) := \sum_{x=0}^{N_A-1} \left(I(x_A + x) - I(x_B - d + x)\right)^2 \quad ,(2)$$

where $N_A = 20$ is the width of a window taken from the inner image, x_B is the outer image which has zero disparity with respect to x_A (start of inner image). Due to the setup of the imaging mirrors only a one-dimensional correspondence search is needed yielding a disparity range of $N_d = 30$ pixels, i.e. $d \in [0, 29]$. For each estimated disparity $d_{\min,i}$, $i = 0, 1, \ldots, N-1$, we compute a quality value $q \in [0, 1]$ depending on uniqueness and reliability of the found match.

After the stereo computation, the current place can be

represented in memory by $N = 72$ disparities and their corresponding quality values[1], $(\mathbf{d}, \mathbf{q}) = \{(d_i, q_i), i = 0, 1, \ldots, N-1\}$, which we call a "disparity signature" of the considered location.

Using elementary trigonometry, distances r to surrounding objects can be computed according to

$$r(d) \approx \alpha/d - r_0 \ , \quad \alpha \approx 2100\,\text{mm} \times \text{pixel} \ , \quad (3)$$

where $r_0 = 29.5\,\text{mm}$ is the distance between the virtual nodal points (A,B) and the robot axis (see Fig. 1 b).

2.2 Homing algorithm

By comparing the current signature with a stored one, it should be possible to return to the place where the signature has been memorized within a certain neighborhood. To investigate this we have extended the homing algorithm described in [7] for the use of disparities:

Using the current disparity signature $(\mathbf{d}, \mathbf{q})$, we compute for several possible movements of the robot (rotations about an angle φ followed by a straight move of length l) predicted signatures $\{(\mathbf{d}^{\text{c}}(\varphi_i, l_i), \mathbf{q}^{\text{c}}(\varphi_i, l_i)), i = 0, 1, \ldots N_{\text{c}}-1\}$ using (3) and trigonometric calculus. Occlusions are dealt with by setting the corresponding quality values to zero. To avoid wrong disparity predictions due to uncertain disparities (low quality value) we have excluded disparities with $q < 0.7$. The similarities of the predicted signatures to the stored signature at the home position, $(\mathbf{d}^{\text{h}}, \mathbf{q}^{\text{h}})$, are estimated according to

$$E^{\text{d}}_{\text{h}}(\varphi_i, l_i) = \min_s \frac{\sum_{k=0}^{N-1} q^{\text{h}}_k \, q^{\text{c}}_{k_s}(\varphi_i, l_i) \left(d^{\text{h}}_k - d^{\text{c}}_{k_s}(\varphi_i, l_i)\right)^2}{\sum_{k=0}^{N-1} q^{\text{h}}_k \, q^{\text{c}}_{k_s}(\varphi_i, l_i)} \quad (4)$$

where $k_s := (k + s) \bmod N$, $s = 0, 1, \ldots N-1$. In the current implementation the considered positions ($N_{\text{c}} = 132$), lie on a hexagonal grid within a radius of approximately 10 cm. Subsequently the robot moves to the position $(\varphi_{\text{opt}}, l_{\text{opt}})$, which minimizes (4). We will call $(\varphi_{\text{opt}}, l_{\text{opt}})$ the "homing vector". To reduce influence of single wrong decisions, the covered distance is limited to $l < 5\,\text{cm}$. These steps are repeated until the position of highest similarity deviates only marginally from the current position, i.e. $l_{\text{opt}} < l_{\text{thresh}} = 5\,\text{mm}$.

3 Route learning

In our robot implementation, the recognition-triggered responses consist of pairs of panoramic views and scene-based homing steps [6]. The views can be one-dimensional 360° records of either the grey values at the horizon, or of the stereo disparities of the surrounding objects, depending on the used homing scheme. For simplicity, we use the terms snapshot or view for both types of place signatures in the remainder of the text.

[1] To simplify notation we omit the index 'min' in the following.

The set of snapshots taken to represent a route should satisfy two criteria: First, a large distance should be covered with a small number of snapshots to keep processing requirements small. Second, the spatial distance of neighbouring views should be small enough to allow reliable navigation between them. If one intends to use the learned routes in a topological navigation system, a third criterion has to be added: the views should be distinguishable. In purely view-based routes, this is the only way to guarantee that route integration can be done properly. One way to fulfil this criterion is to incorporate only distinct views into the routes.

The selection of the snapshots is based on the current view and the stored snapshots. The criteria can be fulfilled by measuring the degree of similarity between views: Dissimilar views tend to be distant in space and are distinguishable by definition, and similar views often are spatially close.

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 (containing either grey values or disparities as entries), after first rotating one of them such as to maximize the overlap with the other one. Whenever a threshold of the view 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. 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. 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 use this classifier for both tasks in our topological navigation system (see Sect. 4).

Using this simple classifier, the recording of routes is straightforward. If the view distance of the current view to the stored snapshots exceeds a threshold value, the robot takes a new snapshot and connects 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. After having taken a snapshot, the robot has to decide where to go next. 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.

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 routes tend to concentrate in the open space between obstacles.

```
REPEAT {
  compute view distance d of current
    view to all snapshots
  read out proximity detection
  IF no obstacle AND d < threshold THEN
    move into current exploration direction
  IF no obstacle AND d > threshold THEN {
    take new snapshot
    choose new exploration direction
  IF obstacle THEN
    modify exploration direction
  }
UNTIL dead end reached OR
    maxtime between snapshots exceeded
```

After a route has been recorded, using it for route navigation is again straightforward as the route consists of a chain of recognition-triggered responses: starting from the first snapshot in the route, the robot tries to find the next snapshot in the route by scene-based homing. As soon as the current view becomes sufficient similar to the goal snapshot, this event triggers a homing run to the next snapshot in the route as goal. This procedure is continued until the last snapshot of the route is reached.

4 View graph

As we said in the beginning, a topological navigation system needs the capability of route integration to form a graph-like representation of the environment. In our case, detecting whether two routes run through the same place amounts to detecting identical views in two different routes. This, however, can only be done if all recorded views are unique. In our system, this is ensured by the view classifier which allows only sufficiently distinct snapshots to be recorded. The resulting graph-like representation is termed *view graph* [15] with snapshots as vertices and connections traversable by scene-based homing as edges.

In principle, connecting two routes whenever two views are sufficiently similar would be enough for route integration. This, however, turned out to be sensitive to false positives since the views at the low resolution used by the robot tend to be similar in several places in the toy house arena. Therefore, we resorted to a more cautious strategy: Whenever the view distance between the current view and an unconnected snapshot drops below a threshold, the robot decides to home to this snapshot. If homing is successful, route integration is performed, i.e., a newly learnt edge is included into the graph. In cases where the robot gets lost or bumps into obstacles, we start a new exploration run, 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 routes.

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 after a successful route integration: The system determines the directions of all neighbouring vertices and directs the next exploration step to the largest open angle. In addition, we use a several other routines that basically limit the connectivity of the vertices and prevents intersection of edges. This leads to an exploration behaviour that tends to concentrate on the least explored regions of the view graph, i.e., regions with a smaller number of snapshots and less connections between them. Further details can be found in [6].

The main loop of the route learning algorithm has to be expanded accordingly:

```
REPEAT {
  :
  compute view distance d2 to all
    unconnected snapshots
  IF no obstacle AND d2 < threshold THEN {
    home to snapshot
    IF vertex reached THEN {
      connect routes
      compute new exploration direction
    }
    ELSE start new graph
  :
  }
UNTIL ...
```

For using the recorded view graph for topological navigation, one needs an additional planning module that can generate routes between a chosen starting view and a goal view. This can be achieved by standard graph search algorithms, e.g., as described in [15]. The generated routes can be navigated by using the route navigation module described in the last section.

The recorded view graphs typically contained 20 to 50 snapshots and 30 to 60 edges, covering about two thirds

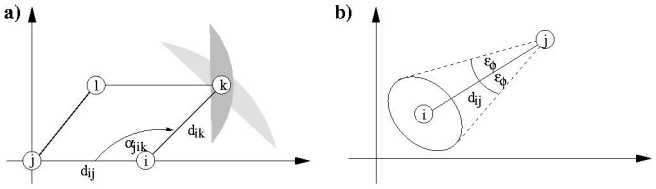

Figure 3: **a)** Position estimates for k are affected by accumulated errors along the paths {j,i,k} and {j,l,k}. Calculating the position for k with respect to the whole graph reduces the position error of k. **b)** Due to the embedding procedure, the position error tends to be homogeneous over the graph. As a consequence, the angular error ϵ_ϕ must decrease proportional to d_{ij}^{-1}

of the toy house arena. Since we required the snapshots to be distinguishable, a single graph could cover only areas with unambiguous view information. This general problem of topological navigation is known as *perceptual aliasing* [13]. One way to cope with this problem is to use context information, e.g., by embedding the view graph into a metric map with the help of additional metric information from path integration. This leads us to the final layer of the navigation hierarchy: survey navigation.

5 Metric embedding of a view graph

A possible way to distinguish between similar views seen at different locations is to label the snapshots with their respective recording positions. The consistent embedding of this position information into a global metric map gives the agent the ability to perform survey navigation, i.e. the agent is able to find shortcuts apart from the learned routes. In the following, we assume that the robot collects position information from its odometry in addition to the snapshots, such that each vertex of the view graph contains a snapshot and an odometric position estimate.

If the robot returns to an already known place by scene-based homing, it closes a loop in the graph. Considering the cumulative error in the robot's odometry, it is clear that a simple vector addition will lead to erroneous position estimates along the path and to contradicting position estimates at the starting vertex (see figure 3a). Instead of calculating path integration along single paths we use a graph–embedding procedure which takes all available routes into account and prevents the accumulation of errors [9].

5.1 Multidimensional Scaling

An agent which uses view information and metric relations simultaneously can be modeled by a state vector containing the perceived view I_t, the current position (x_t, y_t) and the current heading (ϕ_t):

$$S_t = (I_t, x_t, y_t, \phi_t) \tag{5}$$

The distance between two vertices, or the length of an edge can be calculated from two successive odometric position estimates stored for two vertices v_i and v_j:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{6}$$

The angle between two edges sharing a common vertex are calculated from three states:

$$\alpha_{jik} = \pi - \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) + \arctan\left(\frac{y_k - y_i}{x_k - x_i}\right) \tag{7}$$

The embedding of a graph V is mathematically equal to finding a function $f(V) \rightarrow I\!R^2$ which assigns a position $\tilde{\vec{x}_i}$ to the node v_i. The result of the graph embedding is a configuration of points $X = (\tilde{\vec{x}_1}, \tilde{\vec{x}_2}, \ldots, \tilde{\vec{x}_n})$, where pairs or triples of nodes should fulfill the geometrical constraints given by (6) and (7).

Such problems can be solved by *multidimensional scaling* methods [12]. A closed form solution exists if all pairs of distances are known, i.e. if the graph is fully connected [12]. Instead of using all distances contained in the graph, which is computationally prohibitive in larger graphs, we use the additional angle information. The resulting point configuration is unique if the initial path integrator state $S_0 = (I_0, 0, 0, 0)$ is used to setup the coordinate system for all position estimates, i.e. $\tilde{\vec{x}_0} = (0, 0)$ and $\tilde{\vec{x}_1} = (\tilde{x_1}, 0)$. Finding an appropriate point configuration is an optimization problem with two error functions. The first term describes the mismatch error in the distance judgements

$$E_d(X) = \sum_{(i,j)|d_{ij}\neq 0} \left(\| \tilde{\vec{x}_j} - \tilde{\vec{x}_i} \| - \frac{2d_{ij}}{\max d_{ij}} \right)^2. \tag{8}$$

and the second term the error of the angular match[2]:

$$E_\alpha(X) = \tag{9}$$
$$\sum_{(j,i,k)|\alpha_{j,i,k}\neq 0} \left\| \frac{\tilde{\vec{x}_j} - \tilde{\vec{x}_i}}{\| \tilde{\vec{x}_j} - \tilde{\vec{x}_i} \|} - R(\alpha_{jik}) \frac{\tilde{\vec{x}_k} - \tilde{\vec{x}_i}}{\| \tilde{\vec{x}_k} - \tilde{\vec{x}_i} \|} \right\|^2$$

Finally, both functions are combined over a weighted sum:

$$E(X) = \lambda E_d(X) + (1 - \lambda) E_a(X), \quad \lambda \epsilon [0, 1] \tag{10}$$

The weighting parameter λ can be used to compensate for systematic errors in the robot's path integrator. In the experiments, λ is kept constant at 0.5.

Given an arbitrary configuration X, function (9) is limited by $E_\alpha(X) \leq 4N$ where N is the number of terms

[2] $R(\alpha_{jik}) \in I\!R^{2\times 2}$ is a rotation matrix

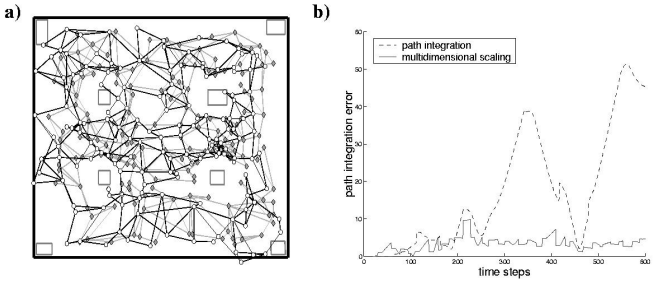

Figure 4: **a)** Resulting map. Vertices marked by diamonds indicate the correct positions. Vertices marked by dots are the position estimations calculated with the MDS–method. Landmarks are shown as gray boxes. **b)** Error in the path integrator position over a time interval. Jumps in the mds–curve show points where the path integrator was recalibrated. Decreases in the path integration curve are only random.

in the sum. Contrary to function (9), function (8) has no upper limit, which could assign an excessive weight to the distance error term. This problem is solved by normalizing the measured edge distances by $\frac{1}{2}\max(d_{ij})$ such that the maximal distance equals 2. Selecting a start configuration randomly in a circle around the origin with a radius of $2|V|$ guarantees a solution for which function (8) and (9) are equally weighted. The resulting configuration is scaled up again by $\frac{1}{2}\max(d_{ij})$.

5.2 Application to large graphs

The optimization problem defined by (10) has a dimensionality of $2|V|-3$. With a growing number of vertices a direct application of the MDS–method becomes impractical. Therefore the following heuristic is used to calculate the position estimations in real–time:

1. A subgraph around the current location v_c is selected containing all vertices which are less than ϵ edges away from the current vertex: $G' \subseteq G$ with $V' = \{v | d(v, v_c) \leq \epsilon\}$.
2. The MDS–procedure embeds G' into a local coordinate system with $\vec{x_c}' = (0,0)$ as the origin, resulting in a point configuration X' for the subgraph.
3. G' is merged back into the global map G. The coordinate transformation is described by a rotation R and a translation T which minimizes the position difference between the corresponding point sets X and $RX' + T$. For this problem a closed form solution exists (see e.g. [10] for a detailed description).
4. The new position estimates are combined with older ones using a simple temporal lowpass filter in order to make the global map more stable. This becomes necessary especially as the subgraph selection could break graph loops.

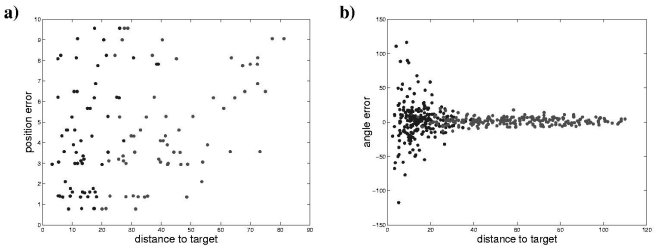

Figure 5: The plots show that the MDS–method bounds the position error over the whole environment. Therefore the map could be used to plan paths between arbitrary pairs of vertices. **a)** Position error of the target vertex $\vec{x_g}$ with respect to the distance from start vertex $\vec{x_s}$. **b)** Error in the calculated driving direction ϵ_α with respect to the distance.

5.3 Experiments

For the experiments we use a simulation of a khepera–robot in the toy house arena. The simulated path integrator was able to measure rotations and translations with a precision of $\pm 10\%$. The heading of the agent is assumed to be known within the visual resolution of the panoramic image, i.e. $\pm 2.5°$ with 72 pixels.

Exploration is done in the same way as in topological navigation. In addition to the view distance, a metric distance to the recorded snapshots $d_M(\vec{x_t}, \vec{x_i}) = \min_i ||\vec{x_t} - \vec{y_i}||$ is calculated at each time step. The classifier which decides when to take a new snapshot now also takes the metric distance into account. If the view distance is below a certain threshold and d_M is greater than 5cm a new vertex is added and the agent selects a new exploration direction by rotating about a fixed angle of 90°. As before, route integration is performed by homing when both view and metric distance fall below a certain threshold. After a successful homing run, a new edge is added. Finally, the position estimates are updated with the MDS–algorithm and the path integrator is recalibrated to the improved position estimation.

In the pseudocode examples given above, survey navigation leads to a further expansion of the main loop (besides complementing the view classificator with metric distances):

```
REPEAT {
  :
  IF no obstacle
     AND view distance < threshold1
     AND metric distance < threshold2 THEN {
    home to snapshot
    IF vertex reached THEN {
      connect routes
      compute new exploration direction
      update all positions using MDL
      recalibrate path integrator
```

```
    }
  :
  }
UNTIL ...
```

The resulting graph is shown in figure 4a. The map quality is measured by the agent's ability to navigate reliably over larger distances apart from learned routes. Given a start vertex v_s and a target vertex v_g, the direction to the goal is calculated by $\alpha = \arctan(y_s - y_g)/(x_s - x_g)$. If the error $\vec{\epsilon_i}$ in the position estimates is approximately homogeneous over the graph, i.e. $\forall_i : ||\vec{\epsilon_i}|| \approx const.$ the error in the calculated driving direction must decrease proportional to the distance $\epsilon_\alpha \sim ||\vec{x_s} - \vec{x_g}||^{-1}$ (see figure 3b), which is indeed the case for the resulting map (see figure 5a and 5b).

6 Concluding remarks

In the above section, we have presented all building blocks for a biomimetic survey navigation system, from scene-based homing as local navigation method to metric embedding of view graphs. All levels up to topological navigation have been implemented on mobile robots, only the final level, survey navigation, still runs only in a Virtual Reality environment.

The system is biomimetic in the sense that all behaviours of the navigation hierarchy are implemented in a bottom-up manner, such that each level of the hierarchy relies on the capabilities of the lower levels. Moreover, all the implemented behaviours can be observed in nature, although the biological algorithms and neural implementation will be certainly different from ours. In this sense, our work is not intended to model a specific animal, but to test whether the hierarchical layering of navigation behaviours can lead to a functional robot navigation system.

Although it is still a long way until such a biomimetic system can be used in technical applications, there are some features that might be interesting also from an engineering point of view: First, simple behaviours tend to be robust with respect to non-stationary environments and sensor errors. This inherent robustness is propagated in a certain sense to the higher layers since these are based on them and do not add low level behaviours on their own. Second, the lower layers provide a backup solution when higher levels fail. For instance, when the global metric map becomes incorrect, the robot still can use the graph structure to find its goal.

Acknowledgements

The authors wish to thank the workshop organizers for giving us the opportunity to participate. Financial support was provided by the Max-Planck-Gesellschaft and the Deutsche Forschungsgemeinschaft.

References

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

[2] K. Cheng. A purely geometric module in the rat's spatial representation. *Cognition*, 23:149–178, 1986.

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

[4] T. S. Collett, B. A. Cartwright, and B. A. Smith. Landmark learning and visuo-spatial memories in gerbils. *J. Comp. Physiol. A*, 158:835–851, 1986.

[5] M. O. Franz and H. A. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133 – 153, 2000.

[6] M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Bülthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5:111 – 125, 1998.

[7] M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biol. Cybern.*, 79:191 – 202, 1998.

[8] L. Hermer and E. S. Spelke. A geometric process for spatial reorientation in young children. *Nature*, 370:57–59, 1994.

[9] W. Hübner and H. A. Mallot. Integration of metric place relations in a landmark graph. In J. R. Dorronsoro, editor, *Artificial Neural Networks - ICANN 2002*, LNCS 2415, pages 825 – 830, Heidelberg, 2002. Springer.

[10] F. Lu and E. E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. Intelligent and Robotic Systems*, 18:249 – 275, 1997.

[11] H. A. Mallot. Behavior-oriented approaches to cognition: Theoretical perspectives. *Theory in Biosciences*, 116:192 – 220, 1997.

[12] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis*. Academic Press, London, 1979.

[13] C. Owen and U. Nehmzow. Landmark-based navigation for a mobile robot. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 5*, Proc. SAB 98, pages 240 – 145, Cambridge, London, 1998. MIT Press.

[14] T. J. Prescott. Spatial representation for navigation in animats. *Adaptive Behaviour*, 4:85 – 123, 1996.

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

[16] W. Stürzl and H. A. Mallot. Vision-based homing with a panoramic stereo sensor. In H. H. Bülthoff, S.-W. Lee, T. A. Poggio, and C. Wallraven, editors, *Biologically motivated computer vision*, Proc. BMCV 2002, LNCS 2525, pages 620 – 628, Heidelberg, 2002. Springer.

[17] O. Trullier, S. I. Wiener, A. Berthoz, and J.-A. Meyer. Biologically based artificial navigation systems: review and prospects. *Progress in Neurobiology*, 51:483 – 544, 1997.