Proceedings 7th Intl. Conf. Artif.
Neural Networks, ICANN-97,
Lausanne, October 1997

The View–Graph Approach to Visual Navigation and Spatial Memory

Hanspeter A. Mallot, Matthias Franz, Bernhard Schölkopf
and Heinrich H. Bülthoff

Max-Planck-Institut für biologische Kybernetik
Spemannstr. 38, 72076 Tübingen, Germany

Abstract. This paper describes a purely visual navigation scheme based on two elementary mechanisms (piloting and guidance) and a graph structure combining individual navigation steps controlled by these mechanisms. In robot experiments in real environments, both mechanisms have been tested, piloting in an open environment and guidance in a maze with restricted movement opportunities. The results indicate that navigation and path planning can be brought about with these simple mechanisms. We argue that the graph of local views (snapshots) is a general and biologically plausible means of representing space and integrating the various mechanisms of map behaviour.

1 Introduction

In animal navigation, three basic mechanisms of spatial memory have been identified:

• *Path integration* or dead reackoning is the continuous update of the egocentric coordinates of the starting position based on instantaneous displacement and rotation data (see [8] for review). Odometry data are often taken from optic flow but other modalities such as proprioception (e.g., counting steps) may be involved as well. Since error accumulation is a problem, the use of global orientation information ("compasses", e.g., distant landmarks or the polarization pattern of the skylight) is advantageous. Path integration involves some kind of working memory in which only the current "home–vector" (coordinates of starting point) is represented, not the entire path.

• *"Piloting": Approaching a place whose local position information matches a stored "snapshot".* This mechanism requires long–term storage of the local position information, such as a view or snapshot visible at that point. From a comparison of the stored view with the current view, an approach direction can be derived. Moving in this direction will lead to a closer match between the two views [1, 4].

• *"Guidance": Associations of recognized views (local position information) to movements.* Here, long–term memory of the local position information (view) is required as well. When recognized, it triggers an action, i.e. a movement or a behavioural routine. The existence of such associations has been shown in bees [3] and humans [7].

Using these basic mechanisms, different levels of complexity of spatial knowledge and behaviour can be formulated. Concatenating individual steps of either piloting or guidance results in routes. These routes will be stereotyped and could be learnt in a reinforcement scheme. More biologically plausible, however, is instrumental learning, i.e., the learning of associations of actions with their expected results. This can be done step–by–step without pursuing a particular goal (latent learning). Instrumental learning entails an important extension of the two view–based mechanisms in that the respective consequences of each of a number of possible choices (either movements or snapshopts to home to) are learnt. This offers the possibility of dealing with bifurcations and choosing among alternative actions. Thus, the routes or chains of steps can be extended to actual graphs which are a more complete representation of space, or cognitive map [10, 12, 11, 5]. The overall behaviour is no longer stereotyped but can be planned and adapted to different goals.

In the project reviewed in this paper, we have explored the mechanisms of visual piloting and guidance with an autonomous robot and combined them into a graph–type cognitive map. A neural network implementation has been developed for the graph of views and movements while piloting has been implemented by conventional algorithms. Experiments with the integrated exploration and navigation system are presented in Sect. 5.

2 Piloting: Matching Current View to Stored Snapshot

Local position information is any sensory input occuring at a place. If a location is not marked by salient features, it has to be defined by *relational* properties, e.g. by an array of surrounding landmarks. The return direction after a displacement can be inferred by comparing the current visual input to the stored snapshot: image regions in the direction of the displacement are expanded while the image in the goal direction is contracted. This requires a matching of the stored snapshot to the current view, i.e., a solution of the correspondence problem [1]. To infer the exact return direction from the pattern of contraction and expansion, some knowledge of the object distances is also required. In our approach, we neglect the directional variation of object distance. Movement is generated in the direction of maximal image contraction which is a reasonable approximation of the ideal approach direction. In fact it can be shown mathematically that the system will approach the goal with arbitrary accuracy, despite the error introduced by the constant distance assumption [4]. Each stored snapshot has a limited "catchment area" of points from which the approach works. The size of these catchment areas depends on the local variation of the views.

The direction of maximal contraction is estimated by a set of matched filters representing the expected "disparity fields" for a number of movement directions. From a population of these predefined disparity fields and their respective degrees of match to the actual disparities the true displacement direction can be determined.

We use a 360° imaging device consisting of a camera pointing upwards and a conic mirror mounted on top of the camera (see Fig. 2 and [2]). The ring of pixels imaging horizontal rays is called the horizontal ring. In all experiments reported here, the image along this horizontal ring is used, with some prior lowpass filtering. In this situation, both image and matched filters are one–dimensional.

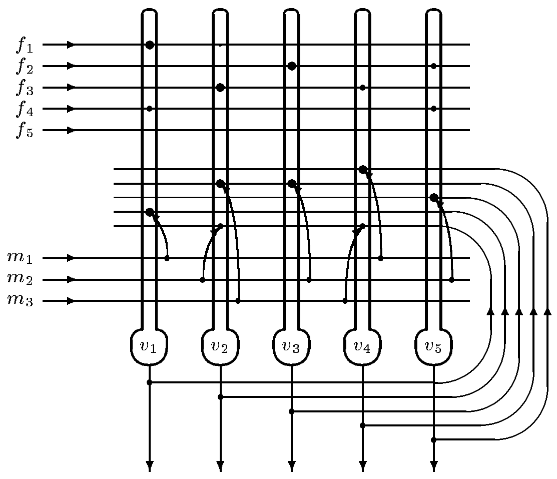

Fig. 1. Wiring diagram of the neural network. $(f_1, ..., f_J)$: feature vector corresponding to the current view. $m_1, ..., m_K$: movement units. $v_1, ..., v_N$: view units. Input weights ϱ_{nj} $(f_j \rightarrow v_n)$ subserve view recognition. Map layer weights α_{ni} $(v_i \rightarrow v_n)$ represent connections between views. They can be modified by facilitating weights $\beta_{k,ni}$ indicating that view v_n can be reached from v_i by performing movement m_k.

3 Guidances: Associations of Views to Movements

A *cognitive map* is a neural mechanism supporting navigation and orientation tasks much as a real map of the environment. In [12], we presented a mechanism for the learning of a cognitive map of a maze from the sequence of local views encountered when exploring the maze. In this approach, the topological structure of a maze is represented as a graph where the places are the nodes and the directed corridors are the edges. The exploration sequence of encountered views corresponds to the sequence of directed corridors travelled along the way through the maze; it can be conceived of as a walk on the **view graph**, a graph whose nodes represent the encountered views and whose directed edges, labelled by movements, represent possible view sequences. Assuming that each corridor corresponds to exactly one view and all views are distinguishable, one can prove that the view graph contains all the information required to reconstruct the place graph.

A **neural network** implementation of the view–graph algorithm (Fig. 1) acquires information concerning three problems:
Identification of views. After learning, perception of a view will be represented by activity in the associated map unit: the unit whose input weights ϱ_{nj} are most closely tuned to the presented view (the "winner" unit).
Learning the maze topology. This is accomplished by developing weights α_{ni} connecting winner units of subsequent time steps within the map layer.
Learning movements. To support path planning, the network must store knowledge about which movement decisions are necessary to generate certain view sequences. This is done by developing modulatory connections from movement units to map layer connections. During path planning, possible motion decisions are presented to the network one after the other. For each decision, the network determines the unit representing the respective neighbouring view, which is evaluated in terms of the time it takes for an activation of this unit to reach the goal view unit.

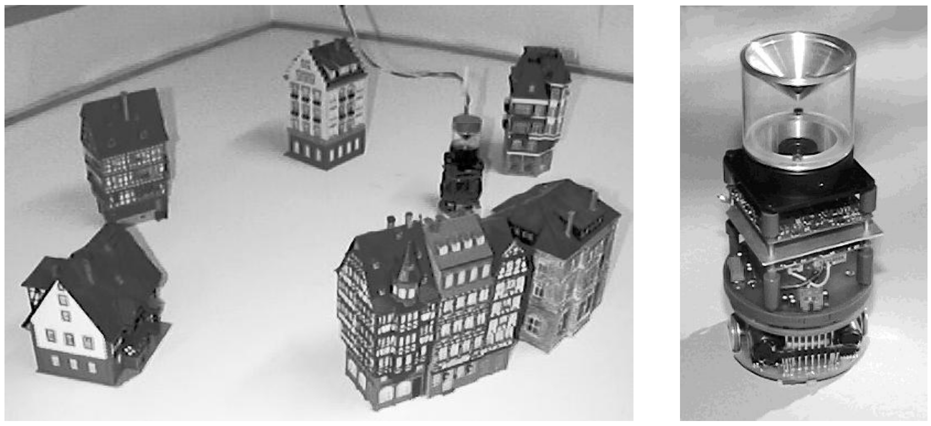

Fig. 2. Left Arena for robot experiments using toy houses. **Right** Close–up of modified Khepera® robot with a vertically mounted camera facing a conic mirror.

4 Exploration: Learning View–Graphs in Open Environments

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 an external structure on the view graph, we have to *select* a set of representative views (referred to as *snapshots*).

The set of snapshots has to satisfy two criteria: First, the views should be distinguishable. In purely graph–based maps, stipulating that the graph nodes be distinguishable is the only way to guarantee that specific views can be navigated to. Second, the spatial distance of neighbouring views should be small enough to allow reliable navigation between them.

In our system for exploring open environments [5], the nodes of the view graph are identified with one–dimensional 360° snapshots of the surrounding panorama (see Sect. 2). We use a simple threshold classifier for selecting snapshots. Whenever the pixel–wise cross–correlation between current view and stored snapshots drops below a threshold, the system takes a new snapshot and links it to the last one. The threshold is chosen such that the nodes remain in the catchment areas of their neighbours. In this way, the system can navigate between views using the visual piloting procedure described in Sect. 2.

This route learning procedure has no means of forming new links to previously visited views, i.e. the resulting graphs would be mere chains. Nontrivial graphs can be learnt by using the same threshold classifier to detect places where the recorded chains overlap: If the cross–correlation between current view and another node exceeds the threshold, the system decides to approach this node by piloting (link verification). If the approach is successful, a newly learnt link

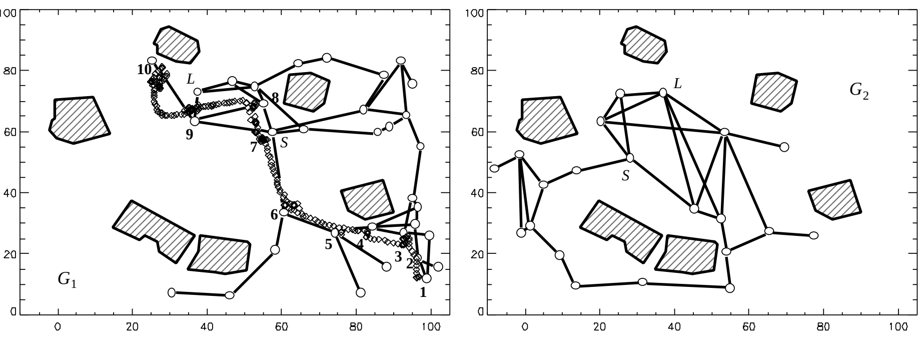

Fig. 3. Two view graphs in a 118 × 102 cm size arena with different start positions S. Locations of snapshots are marked by circles (∘), the lines show recorded links between them. The sample trajectory marked by diamonds (⋄) in G_1 started at node 1. Node L is a possible linking place between G_1 and G_2. Shaded areas are obstacles.

is included into the graph. In cases where the system gets lost, a new graph is initiated, which will typically get connected to the old one in due course.

5 Robot experiments

The experiments were conducted both in a hexagonal maze and in an arena cluttered with model houses, using a modified Khepera® platform. In the maze experiments, visual cues were provided by one–dimensional patterns on the floor, read with two infrared sensors during traversing the corridors ("two–pixel vision"). In the open arena, no artificial landmarks were used; the imaging system on the robot comprised a conical mirror mounted above a small video camera pointing up to the center of the cone as described above (Fig. 2; [2]).

The maze navigation system was tested in different behavioural modes. In **exploration mode**, the robot was driving through the maze (about 100–200 randomly selected movement decisions) using its obstacle avoidance to get around. After the network had been trained on these movements and views, it was tested whether maze views and topology are correctly encoded in the network connectivity graph by measuring combinatorial distances of winner units of subsequent time steps. The measured "neighbourhood preservation rates" (the map quality criterion used by [12]) were close to 100%, showing that after learning, the robot "knew" its position in the maze. The network generates *expectations* about which views are likely to be seen next, given the previous view and the movement carried out. Once the map has been learnt, this makes the system rather robust against noise. In **navigation mode**, the robot used the information stored in the neural network to find a particular view (the "goal" view) from various starting positions. This goal view may be any view in the maze, without additional learning required. In our small experimental mazes, the robot usually found the shortest way to the goal view. This even works "in the dark", without any view input (see [12, 6]) mimicing a path–integration mechanism (as the one suggested by [9]).

Fig. 3 shows two examples of view graphs G_1 and G_2 in an open environment. Since the system only records snapshots which are sufficiently distinguishable, the number of snapshots is limited. As a consequence, systems like ours that use only topological information during exploration are necessarily confined to a subregion of the arena, where the visual information remains unambiguous. In order to cover the entire arena, local graphs such as G_1 and G_2 in Fig. 3 can be combined. Common nodes between subgraphs have to be marked as *linking places* to allow transitions between them. In our example, node L is common both to G_1 and G_2 and could be used to switch from one subgraph to the other.

Once the graph has been learnt, one can generate a path to a goal by standard graph search algorithms, and then sequentially navigate along this path by piloting. The usefulness of the view graph for path planning is demonstrated by the sample trajectory in G_1. The robot traverses a chain of 10 nodes, thus connecting regions which have no visual overlap.

In future work, we plan to integrate all three basic mechanisms of navigation into a joint, graph–based architecture.

References

1. B. A. Cartwright and T. S. Collett. How honey bees use landmarks to guide their return to a food source. *Nature*, 295:560 – 564, 1982.
2. J. S. Chahl and M. V. Srinivasan. Visual computation of egomotion using an image interpolation technique. *Biol. Cybern.*, 74:405 – 411, 1996.
3. T. S. Collett and J. Baron. Learnt sensori–motor mappings in honeybees: interpolation and its possible relevance to navigation. *J. comp. Physiol. A*, 177:287 – 298, 1995.
4. M. O. Franz, B. Schölkopf, and H. H. Bülthoff. Homing by parameterized scene matching. In *Proc. 4th Europ. Conf. on Artificial Life (to appear)*, 1997a.
5. M. O. Franz, B. Schölkopf, P. Georg, H. A. Mallot, and H. H. Bülthoff. Learning view graphs for robot navigation. In *Proc. 1. Intl. Conf. on Autonomous Agents*, 1997b.
6. H. Mallot, H. Bülthoff, P. Georg, B. Schölkopf, and K. Yasuhara. View–based cognitive map learning by an autonomous robot. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings ICANN'95*, vol II, pp 381–386. EC2, Nanterre, France, 1995.
7. H. A. Mallot and S. Gillner. Psychophysical support for a view–based strategy in navigation. *Invest. Ophth. Vis. Sci. Suppl.*, 38, 1997.
8. R. Maurer and V. Séguinot. What is modelling for? A critical review of the models of path integration. *J. theor. Biol.*, 175:457 – 475, 1995.
9. B. L. McNaughton, C. A. Barnes, J. L. Gerrard, K. Gothard, M. W. Jung, J. J. Knierim, H. Kudrimoti, Y. Qin, W. E. Skaggs, M. Suster, and K. L. Weaver. Deciphering the hippocampal polyglot: The hippocampus as a path integration system. *J. experimental Biol.*, 199:173 – 185, 1996.
10. B. Poucet. Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms. *Psychological Rev.*, 100:163 – 182, 1993.
11. T. Prescott. Spatial representation for navigation in animals. *Adaptive Behavior*, 4:85 – 123, 1996.
12. B. Schölkopf and H. A. Mallot. View–based cognitive mapping and path planning. *Adaptive Behavior*, 3:311 – 348, 1995.