Mapbuilding using Self-Organising Networks in "Really Useful Robots"*

Ulrich Nehmzow and Tim Smithers†
Department of Artificial Intelligence
University of Edinburgh
5 Forrest Hill
Edinburgh EH1 2QL
Scotland
Telephone 031 - 667 1011 ext 2529/2517
Telex 727442 UNIVED G
E-mail:ulrich@uk.ac.ed.edai tim@uk.ac.ed.edai

September 1990

Abstract

The Really Useful Robots (RUR) project is seeking to understand how robots can be built that develop and maintain the task achieving competences they require for flexible and robust behaviour in variable and unforeseen situations, as opposed to these being installed by their designers. In this paper we present an experimental autonomous robot with a map building competence which uses a self-organising network. Map building forms a necessary step on the way to development of a navigational competence. Some encouraging initial test results are also presented.

1 Introduction

The traditional approach to control in (mobile) robots is to decompose the task into separate components, and implement these using standard control techniques, see [Levi 1987], for example. This we call an *analytical approach*. Alternatively, a control structure can be built 'bottom up', first building foundational competences (such as 'move around and avoid obstacles'), and later on top of these more complicated competences (such as 'explore', 'map building', and 'map using'). This we call a *synthetic approach*, see [Brooks 1986], for example.

At Edinburgh we have adopted a synthetic approach in what we call the 'Really Useful Robots' project (RUR) [Nehmzow et al 1989]. This project is attempting to develop a control architecture which supports the development of task achieving competences by the robot. In other words, we are trying to understand how a robot can sequentially acquire and maintain the behavioural competences it requires, rather than have them 'installed' by us as its designers. We believe that this *autonomous acquisition* of task achieving competences will lead to greater flexibility and robustness in the behaviour of robots with respect to variable and unforseen situations. In investigating this idea we are motivated and informed by the adaptive control mechanisms we see in simple animals which result in them having flexible, reliable, and robust competences well matched to the tasks they are responsible for achieving and to the environment in which they are exercised.

Trying to get a robot to acquire the skills it needs means that as many decisions as possible are left to the robot, rather than being predefined by the designer. *Alder*, the first of the 'Really Useful Robots' (see figure 6)[1], is able to adapt to a changing environment, and to acquire useful competences. It uses what we call

 — Reprint from J.A. Meyer and S. Wilson (eds.), *From Animals to Animats*, Proc. 1st conference on Simulation of Adaptive Behaviour, MIT Press 1991

†Names appear in alphabetical order, with both being principal authors on this occasion.

[1] Alder is a mobile robot whose base is built with a Fischertechnik kit. It is about 25cm long, has an 8052 based microcomputer on board (16k RAM) and is equipped with up to seven tactile sensors plus odometer. In addition to this a sonar sensor is available, but has not been used to obtain the results presented in this paper. More information about Alder and the 'Really Useful Robots' approach can be found in [Nehmzow et al 1989].

fixed and *plastic* components in its control architecture to achieve this, fixed components being the mechanics of the robot, its shape (morphology), the control program, and its so-called 'instincts', the plastic component being an artificial neural net (in the current system). So far we have successfully demonstrated robot acquisition of simple obstacle-avoidance, dead-end-escape, and wall-following competences. The next stage is to get the robot to develop a navigational competence, using a similar approach. This first requires a mapbuilding competence.

As in *controlling* mobile robots, so in *mapbuilding* a traditional approach can be taken. The traditional approach either uses some kind of fixed structure that is filled with information as it is obtained by sensory activities,see, for example, [De Almeida 1989], [Moravec 1988], [Goto and Stentz 1987] or [Tsai et al 1990]. We call these 'self detailing schemes'. Alternatively, a complete 'ready-made' map may be provided from the start, see, for example, [Tsubouchi and Yuta 1987], which we call 'pre-installed map schemes'. In these traditional approaches the structure of the map is defined by the *designer*, and so is not necessarily the best to store and retrieve the knowledge the *robot* needs to have and can acquire about the world. Here, as well as in controlling robots, *predefining means limiting*. Limiting what is represented by a map is not of itself a problem, but knowing what needs to be represented and what doesn't is. To get around this problem we think that the process of constructing a map should be left, as far as possible, to the robot, not to the designer, the idea being, that a map built by the robot is likely to be more useful to it, than one given to it by its designer.

Alder uses a self-organising network (see [Kohonen 1988]) to construct internal representations of the world it experiences as it moves around. In this paper we describe how this mapbuilding competence is implemented and some early test results which we believe are encouraging.

1.1 The place of this paper within the Navigational System of Alder

The plan is to design a navigational system as shown in figure 1.

Fixed components (as mentioned above) within this structure are the sensors, including the higher level interpretation of incoming data (the top block in figure 1), and the behaviour[2] of the robot. *Plastic components* are the self-organising network, as described in section 2,

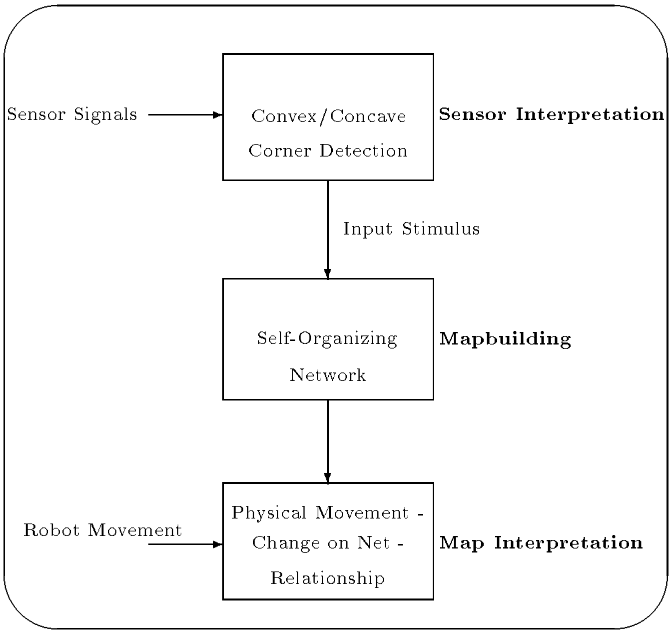

Figure 1: The proposed Navigational System of Alder

and an artificial neural network which associates particular robot movements in the real world with changes in excitation of the self-organising network (the bottom block in figure 1). The combination of all these components forms the navigation system.

This paper discusses only one aspect of this system: mapbuilding, using self-organising networks (the middle block of figure 1).

1.2 What is a map?

We take a map to be *any one-to-one mapping (bijection) of a state-space on to map-space.* Examples of such maps are the plan of the London Underground, the Edinburgh telephone directory, and a family tree. The connections between stations, phone owners and numbers, and family members are represented in all these cases. A map is therefore not just a representation of a 'birds-eye-view' of the world the robot inhabits.

In our RUR project the state-space that is represented using a self-organising network has nothing to do with actual physical locations in the real world, at least not directly. What *is* represented by these maps is the result of previous and present motor-sensory experience of the robot. The motor actions and sensory actions of the robot are tightly coupled, so tightly coupled in fact that they cannot be viewed independently of each other! Therefore the state space represented *does* in the end

[2] 'Behaviour' here means any fixed pattern of motor-sensory activity, it is therefore a 'fixed' component of the robot.

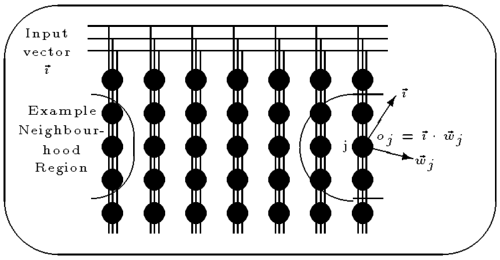

Figure 2: A two-dimensional self-organising network

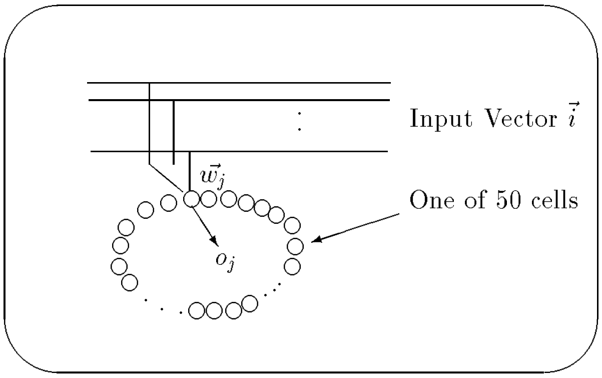

Figure 3: The self-organising network used on Alder

have something to do with actual physical locations of the robot in the real world, but only indirectly through motor-sensory behaviour of the robot.

Whenever the term 'map' is used in this paper it is used in the sense of the definition given above, i.e. a bijection of state-space on to map-space.

2 Self-Organising Networks

Unsupervised learning can be achieved, using self-organising networks [Kohonen 1988]. These networks develop internal representations of the input space by mapping distinct input vectors onto distinct areas of the network (injection). One advantage of these networks is that they can accept redundancy in the input vectors. Redundant information is ignored by the net as long as the input vector contains *sufficient* information for a meaningful mapping to be generated. Surplus information does not impair performance. This fact makes it easier to define a suitable input space. Figure 2 shows an example of a two-dimensional self-organising network.

The input vector $\vec{\imath}$ is the same for all the cells. The output o_j of cell j is determined by the scalar product of input vector $\vec{\imath}$ and the weight vector $\vec{w}_j$ of cell j, and is given by:

$$o_j = \vec{w}_j \vec{\imath} \tag{1}$$

where $\vec{w}_j$ is the individual weight vector of cell j. The weight vectors are unit vectors: $||\vec{w}_j|| = 1$.

Initially, the elements of the weight vectors are set to randomly selected values such that they are all unique. Therefore one cell of the network is bound to respond most strongly to a particular input presented to the net. The weight vector of this maximally responding cell, as well as the weight vectors of all the cells within a defined neighbourhood around this cell, are changed according to the following equation:

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(\vec{\imath} - \vec{w}_j(t)) \tag{2}$$

where η is the so-called 'gain', a value that determines the amount of change (typically around 0.5). Weight vectors outside the specified neighbourhood [3] remain unchanged.

After several 'epochs', i.e. presentations of input vectors to the network, typical dissimilar responses appear for dissimilar input vectors. Thus a mapping arises, whereby different input vectors are mapped onto different regions of the network (bijection). This therefore is a *map* according to the definition given in section 1.2.

2.1 The Self-Organising Network used for Mapbuilding in Alder

Obviously, border effects can occur if the network has edges. This can be avoided by joining the opposite edges of the network. In the case of a two-dimensional network this yields a torus-shaped network.

As long as a network with a sufficient number of cells is used [4] it is not necessary to use a network of the same dimensionality as the space the agent acts in (i.e. although the robot moves in a two-dimensional world, a one-dimensional network can be used to obtain the self-organising state-space-representation). Since the computation for a one-dimensional network is much less [5], and the computational power of Alder is limited, we used a ring of 50 cells as self-organising network (see figure 3).

[3] The neighbourhood size is often chosen to be decreasing over time, in order to achieve quick changes in the beginning, and only little changes later.

[4] A rough calculation to determine the minimum number of cells required: If the network is to represent n distinct states, and the neighbourhood size within which weight vectors of cells are updated is m, then the minimum number of cells obviously is n · m. However, this is the *minimum* number, in practice a bigger number is needed.

[5] The number of necessary weight-vector changes is proportional to n^m, where n is the neighbourhood size and m the dimensionality of the map.

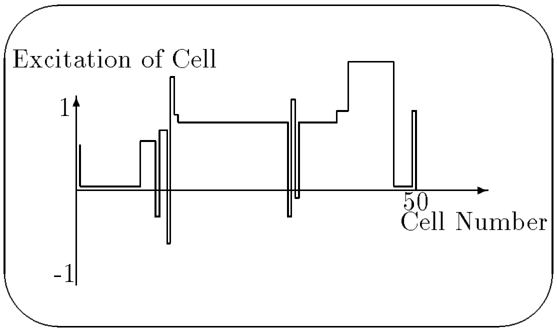

Figure 4: A typical response of the ring after having settled in

The behaviour of this network is as previously described in section 2, the neighbourhood within which weight vectors are updated is ± 2 cells (constant over time). A typical response of the network is shown in figure 4 (in this figure the ring is cut and shown as a line).

The particular response of the network to a particular input stimulus (i.e. the activity pattern of all the cells of the net) can itself be viewed as a vector. This is convenient for later analysis, the particular response vector to a particular vector is called $\vec{\kappa}$ in this paper.

2.2 The Role of Robot Behaviour

Environment, agent (that is robot), and task are very tightly coupled and cannot be treated individually. This became obvious through the following observation:

The first behavioural pattern the robot showed was to wander around randomly and simply avoid obstacles. Distances travelled between obstacles and sensor readings obtained were used as input vectors to the net (see also section 2.3). The result was that the $\vec{\kappa}s$ of the net in the settled state showed no strong correlation to particular physical locations. This is not surprising since if *we* were to wander randomly around a darkened room avoiding obstacles encountered by feeling them, we too would not build up a good 'picture' of the room and what is in it and where.

When we changed the robots behaviour to a wall-following behaviour the results became much more interesting: together with an improved input vector (see 2.3) the $\vec{\kappa}s$ did indeed show a correlation with particular physical locations in the real world. By adopting wall-following behaviour the robot negotiated its way around the perimeter of its enclosure and the $\vec{\kappa}$s corresponded to corners.

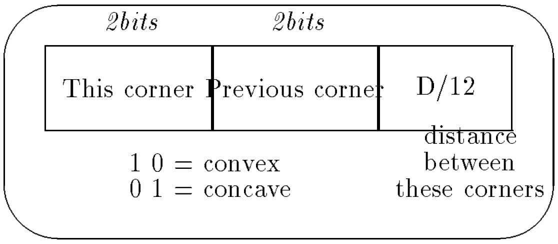

Figure 5: A typical Input Vector

2.3 The Input Vector

A self-organising network clusters the input information presented to it in a (statistically) meaningful way. If the input to the net contains no meaningful information (meaningful in relation to the task of constructing a state-space representation of the robot in its world, that is) the network will not develop any meaningful structure. The very first input vector we used simply contained information about whether the robot had "seen" something on its left or on its right hand side, plus information about two previous sensor readings (again only whether the obstacle was seen on the left or on the right), plus odometry information. Such information is too weak to construct a meaningful state-space representation (at least without requiring a great many input vectors), consequently the $\vec{\kappa}s$ in the settled state of the net had little correlation with particular locations in the real world.

This was obviously due to a lack of sufficient structure in the input vector presented to the net. We therefore enriched the input vector by preprocessing the sensory information obtained. Instead of taking data straight from the sensors and feeding it into the net, sensor readings were used to detect convex and concave corners[6]. This information (whether a corner was encountered, and whether this corner was convex or concave) was then used as input to the net. The input vector eventually used to obtain the results shown in section 3 contained information about the present corner and previous corners encountered as well as the distance travelled between the present and the previous corner (An example of such an input vector is shown in figure 5)[7].

[6] This is very easily achieved: If the time the robot needs to turn towards the wall exceeds a certain threshold time, then it is assumed that a convex corner is detected. Similarly, if the time it takes the robot to get away from a detected obstacle exceeds a certain threshold time, it is assumed that a concave corner is detected.

[7] Note that it is necessary to encode the information about convex and concave corners using 2 bits, because artificial neural networks can only make use of non-zero input lines (see also equation 2).

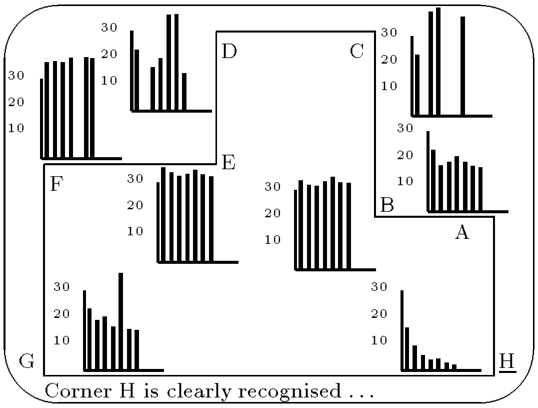

Figure 7: Recognizing corner H, looking at one previous corner

Figure 6: A typical environment for Alder

2.4 A brief summary of the whole mechanism

1. Initialise the self-organising network (i.e. the ring) by filling the weight vectors of all cells with randomly chosen values.
2. Normalise all the weight vectors.
3. Present an input stimulus to the ring (the input stimuli chosen for our experiments are desribed in section 2.3).
4. Determine the response to this stimulus for each cell of the ring according to equation (1).
5. Determine the unit that is responding most strongly.
6. Update the weight vectors of the five units within a neighbourhood of $\pm$ 2 cells of the maximally responding cell, according to equation (2).
7. Normalise those five weight vectors again.
8. Continue with step 3.

3 Experimental Results

The experiment was conducted as follows: The robot was placed within a fence (see figure 6) and allowed to explore it, following the wall for several rounds.

Whenever a convex or concave corner was detected an input vector as described in section 2.3 was created and presented to the ring. The gain η for updating weight vectors (see equation 2) was initially very high (5.0) and decreased by 5 per cent after every presentation of an input vector. The longer the robot went around in the enclosure, the more settled the ring became and the more precise the response to a particular input stimulus. After about three rounds a particular corner was marked by pressing the 'attention' button on the robot. The $\vec{\kappa}$ of the ring for that particular ('marked') corner was then stored ($\vec{\kappa_m}$), all subsequent $\vec{\kappa}s$ were compared with $\vec{\kappa_m}$ by calculating $||\vec{\kappa} - \vec{\kappa_m}||$; the smaller this value, the closer the $\vec{\kappa}$ at that particular corner to $\vec{\kappa_m}$. Obviously, if the robot is able to construct a meaningful internal state space representation this difference should be small when the robot is at the marked corner, and it should be noticeably larger at any other corner.

The input vector chosen will determine the quality of the internal representation obtained. Only information that is actually contained in the input vector sequence can eventually be found on the internal map! The results we obtained highlight this very clearly:

Originally we used an input vector that contained information about the current corner encountered and the previous corner to that one, as well as the distance travelled between these two corners (this vector is shown in figure 5). In figure 7 we can see that corner H is uniquely identified, using this input[8].

It is obvious that corner H is the easiest of all corners to recognize, because of the long distance travelled between corners G and H.

Looking at corners C and F, on the other hand, one

[8] The bars in these diagrams show $||\vec{\kappa_m} - \vec{\kappa}||$. The smaller this value, the closer the response of the net to the response the net showed at the marked corner. The bigger this difference, the more distinct are the marked corner and the corner currently encountered. In other words: the smaller the value, the more alike are marked corner and present corner.

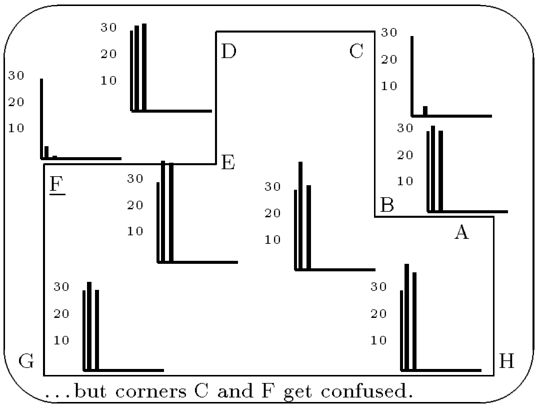

Figure 8: Recognizing corner F, looking at one previous corner

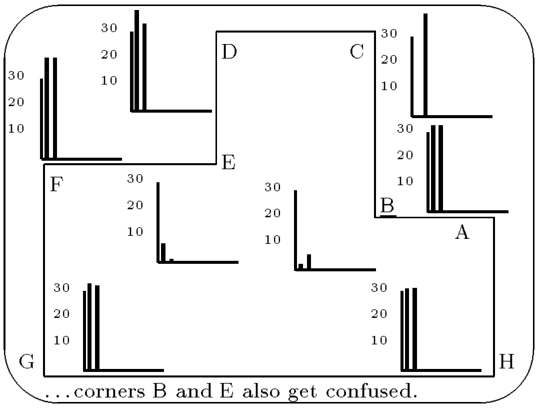

Figure 9: Recognizing corner B, looking at one previous corner

might expect that these corners get confused on the internal map, because both C and F are concave corners, the previous corner in both cases is convex, and the distances between them is similar. Exactly this is observed (see figure 8).

The same applies to corners B and E: they are both convex, both have previous corners that are concave, and again the distances between them are similar. Again, our expectations that these two corners might have an identical representation on the resulting map are experimentally verified (see figure 9). This confusion of corners C and F and B and E respectively arises from the fact that the motor-sensory-space of the robot (input-space of the network) is not well coupled to the physical space the robot is acting in. According to the definition given in section 1.2 the resulting network is indeed a map, but a map of the motor-sensory space rather than the physical space[9].

The obvious answer to this problem is to increase the information contained in the input vector. Taking the example of corners B and E, even looking at *two* previous corners is not going to avoid confusion of these two corners, because for both corner B and E the respective previous two corners are of the same type, and the distances travelled are similar too.

To verify our theory we nevertheless used an input vector that contained information about the present corner as well as *two* previous corners: as expected corners B and E still cannot be distinguished (figure 10), and corners H gives no problems, as before (figure 11).

Only if the input vector contains information about the present corner as well as the *three* previous corners do corners B and E have different input vectors, there-

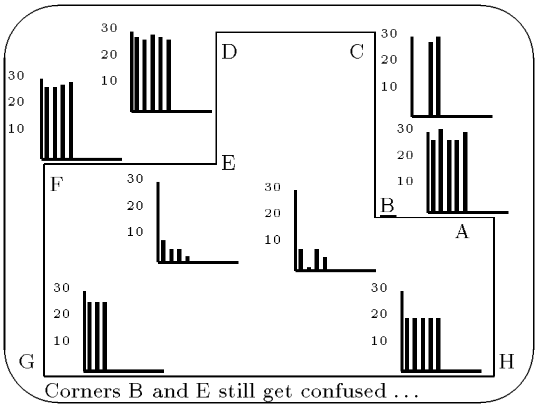

Figure 10: Recognizing corner B, looking at two previous corners

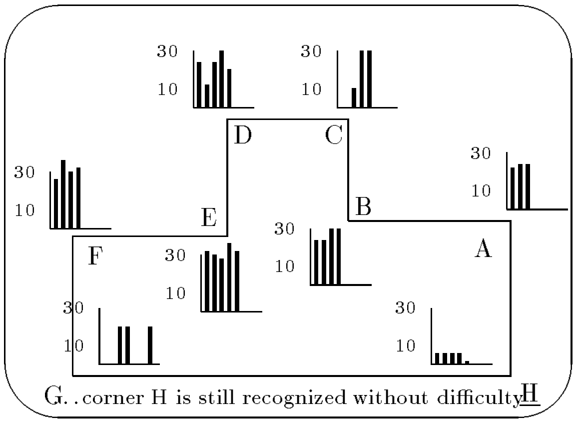

Figure 11: Recognizing corner H, looking at two previous corners

[9] The mapping of the physical space onto map-space here is an injection, that of the motor-sensory space a bijection.

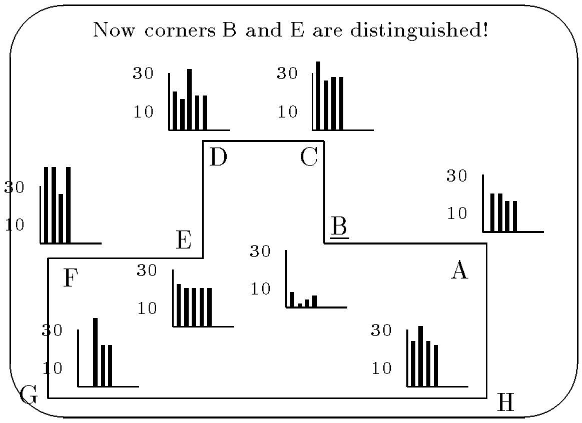

Figure 12: Recognizing corner B, looking at three previous corners

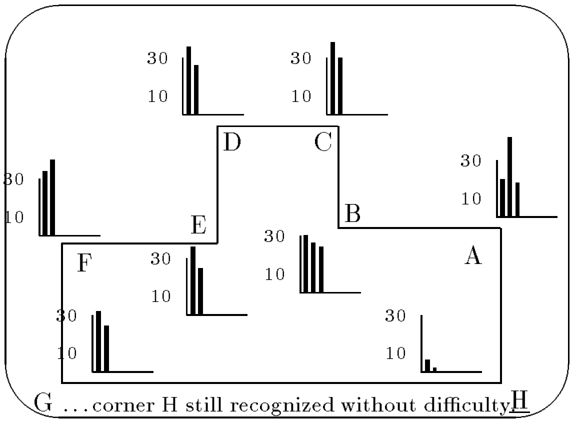

Figure 13: Recognizing corner H, looking at three previous corners

fore one would expect the map to be able to distinguish them. This is exactly what happens (see figure 12)!

Again, as in all the previous cases, corner H is recognized without any problems (see figure 13).

These observations show the relationship between input vector and resulting internal representation very clearly, they also show that self-organising networks *can* be used to generate meaningful internal representations of the world.

Another observation we made is that the 'certainty' increases with increasing experience: $||\vec{\kappa_m}(t)-\vec{\kappa_m}(t+1)||$ decreases continuously as the robot's experience increases (see figure 14).

In summary, this is what happened: Having had sufficient time to explore the environment, the robot was able to recognize particular corners that had been marked by the experimenter, provided this was theo-

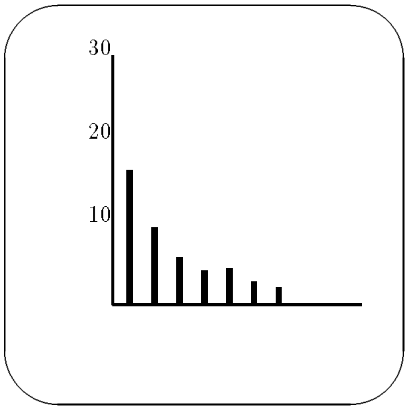

Figure 14: $||\vec{\kappa_m}(t) - \vec{\kappa_m}(t+1)||$

retically possible[10]. In cases where the input vector presented to the self-organising net contained insufficient information, Alder confused corners that looked alike, simply because their input vectors were (almost) identical. This is not surprising. The robot's ability to recognize the marked corner reliably increased with experience, $||\vec{\kappa_m}(t) - \vec{\kappa_m}(t+1)||$, the difference between excitation of the net at the marked corner at time t and excitation at the marked corner at time t+1, decreased continously. It eventually became zero.

4 Comparison with Bee Navigation

As [Cartwright and Collett 1983] have found in their experiments, *Apis mellifera* use nearby landmarks to guide their way to a food source. "[Bees] do not find their way using anything analogous to a floor plan or map of the spatial layout of landmarks and food source. The knowledge at their disposal is much more limited, consisting of no more than a remembered image of what was present on their retina when they were at their destination. ... Bees find their way, the experiments seem to say, by continuosly comparing their retinal image with their snapshot and adjusting their flight path so as to lessen the discrepancy between the two." We do not claim that the mechanism used for landmark learning in Alder is identical to the one used in bees; however, there are similarities. Like bees, Alder does not generate a floor plan (a conventional map), instead it uses 'snapshots' (distinctive excitation patterns of the self-organising network as a response to the sensory input) to recognize locations.

[10] Alder signals the recognition of a marked corner by a swift turn, first to the left and then to the right.

Another similarity to bee navigation is the fact that the mechanism described in this paper is very robust and fairly immune to noise. 'Undetected' corners, varying distance measurements and even moving the robot to a different location (without the control algoithm realising this) do not impair Alder's ability to recognize locations. [Cartwright and Collett 1983]: "The bee's guidance system is immune to a considerable amount of noise".

5 Conclusion

Building an internal state space representation (a *map*) is an essential step towards developing a navigational competence in a mobile robot. We have shown that it can be achieved, using self-organising networks. Particular input stimuli result in distinct reponses of the (ring-shaped) self-organising network, distinct physical locations in the real world can thus be distinguished, provided that robot behaviour and sensing are well coupled to the environment and the task of mapbuilding. The mechanism presented is similar to the 'feature recognition' mechanism used by bees to navigate—bees compare stored retinal patterns to the current retinal pattern to calculate a flight direction vector ([Gould 1988] [Waterman 1989] and [Cartwright and Collett 1983]).

The next step in the RUR project is to develop a network that can learn the relationships between physical movements in the world and observed activity changes in the self-organising network (the bottom block shown in figure 1), which will make true navigation possible.

Acknowledgements

The work reported here is supported by a grant from the UK Science and Engineering Research Council (grant number GR/F/5852.3). We would also like to thank John Hallam and Peter Forster, our colleagues on RUR, and Mitch Harris for helpful comments on earlier drafts of this paper.

References

[De Almeida 1989] R, De Almeida, C. Melin: Exploration of Unknown Environments by a Mobile Robot, in: Proceedings of IAS 2, Amsterdam 1989

[Brooks 1986] Rodney Brooks: A robust layered Control System for a Mobile Robot, MIT AI memo 864, 1986

[Cartwright and Collett 1983] B.A. Cartwright and T.S.Collett: Landmark Learning in Bees, Journal of Comparative Physiology (1983) 151:521-543

[Goto and Stentz 1987] Y. Goto, A. Stentz: The CMU System for Mobile Robot Navigation, in:Proc. IEEE intern. Conference on Robotics and Automation 1987 (p. 99)

[Gould 1988] James L. Gould and Carol Grant Gould: The Honey Bee, Scientific American Library, New York 1988, p. 106

[Kohonen 1988] Teuvo Kohonen: Self-Organization and Associative Memory, Springer 1988

[Levi 1987] Paul Levi: Principles of Planning and Control Concepts for Autonomous Mobile Robots, in: Proc. IEEE intern. Conference on Robotics and Automation 1987 (p. 874)

[Moravec 1988] Hans P. Moravec: Sensor Fusion in Certainty Grids for Mobile Robots, AI magazine Summer 1988 (pp. 61-74)

[Nehmzow et al 1989] U. Nehmzow, J. Hallam and T. Smithers: Really Useful Robots, in Proceedings of IAS 2, Amsterdam 1989

[Tsubouchi and Yuta 1987] T. Tsubouchi, S. Yuta: Map assisted Vision System of Mobile Robots for Reckoning in a Building Environment, in: Proc. IEEE intern. Conference on Robotics and Automation 1987 (p. 1978)

[Tsai et al 1990] Jeffrey J.-P. Tsai, Mark Metea and John Cesarone: A knowledge-based navigation scheme for autonomous land vehicles; in: Applied Artificial Intelligence 4:1-14, 1990

[Waterman 1989] Talbot H. Waterman: Animal Navigation, Scientific American Library, New York 1989, p.183