Using Motor Actions for Location Recognition*

Ulrich Nehmzow
Department of Artificial Intelligence
University of Edinburgh
ulrich@uk.ac.ed.aifh

Tim Smithers
Artificial Intelligence Laboratory
Vrije Universiteit Brussel
tim@be.ac.vub.arti1

Abstract

We present a Behaviour-based mobile robot that is able to autonomously build internal representations of its environment and use these for location recognition. This is done by a process of self-organisation, no explicit world-model is given. The robot uses an artificial neural network for mapbuilding and additional adaptive processes to achieve successful location recognition. The robot reliably recognises locations in the world and can cope with the noise inherent to the real world.

1 Introduction

In an ongoing series of experiments we are using simple mobile robots to investigate mechanisms to support the autonomous acquisition of specified competences. These experiments form a part of our "Really Useful Robots" (RUR) project, the aim of which is to develop adaptive control schemes for autonomous mobile robots which are both flexible and robust with respect to variable and unforeseen situations. So far we have investigated schemes for learning obstacle avoidance, dead-end escape, wall-following and corridor-following competences, [Nehmzow, Hallam & Smithers '89], learning location recognition using sensor-based feature detectors, [Nehmzow & Smithers '91], and learning location recognition using motor command sequences, see [Nehmzow, Smithers & Hallam '91].

Figure 1: Alder, the first of the "Really Useful Robots".

In this paper we present a new location recognition scheme which is not only more robust and reliable, but also simpler than our previous schemes. It again uses information derived from motor commands, rather than directly from sensors, which we believe to be novel in autonomous systems research.

Our previous experiments were done using "Alder", the first of our Really Useful Robots (see figure 1). Alder consists of a chassis built from Fischertechnik, an ARC52 controller which uses an INTEL 8052 eight-bit microprocessor and

*Reprint from F. Varela & P. Bourgine (eds.), Towards a Practice of Autonomous Systems, Proceedings of the first European Conference on Artificial Life (ECAL 91), MIT Press, Cambridge Mass. and London, England 1992.

has an on-board BASIC interpreter, an interface card giving independent control for two motors (forward, reverse, and stop for each motor, but with no feedback of distance travelled or angle turned), and up to eight binary sensor inputs. For most of our experiments we have used two whisker sensors which act as omnidirectional tactile sensors. For the obstacle avoidance, dead-end escape, and wall following we used a behaviour-based controller together with a perceptron-like network to provide the necessary plastic element in the learning mechanism (see [Nehmzow, Hallam & Smithers '89] for more details).

For the location recognition experiments we have been using self-organising networks (see section 2.1). In our first experiment the self-organising process was fed with input vectors derived from a sensor-based feature detector programmed into the robot, [Nehmzow & Smithers '91]. The features that were detected and identified were the convex and concave corners of its enclosure (see figure 4), which it came across as it followed the wall around the internal perimeter. Although this scheme proved to be successful, we were dissatisfied with having to base it upon a hard-wired feature detector—we would prefer not to have to 'tell' the robot so much about its environment. In an attempt to do away with the feature detector we devised a location recognition scheme which uses vectors based upon motor commands. In this scheme an input vector to the self-organising network was constructed each time the motor states changed as a result of a new command. This meant that the number of input vectors produced was significantly larger than in the previous scheme. It also meant that it was harder to derive the information necessary to demonstrate location recognition. In the final scheme we used seven two-dimensional self-organising networks all working in parallel. Essentially this scheme performed a kind of frequency component analysis on the pseudo-periodic sequence of input vectors generated as the robot completed circuits of its enclosure. Although this scheme was successful, and it did remove the need for the explicitly programmed feature detector, we were again dissatisfied with it because of its high complexity and computational cost relative to the previous feature detector-based scheme. We also disliked the fact that the seven thresholds for the seven networks for location recognition had to be set by hand, thus making it dependent on 'magic numbers' which have to be set by the programmer.

This led us to devise a new location recognition scheme which is the subject of the experiments presented here. For this scheme we have used a new robot, 'Cairngorm', the third of the Really Useful Robots[1], shown in figure 2.

Figure 2: Cairngorm, the third of the "Really Useful Robots".

Cairngorm also uses a chassis built from Fischertechnik, but has a more powerful controller based on a Motorola 68000 CPU and having 128 Kbytes of memory. Its motor control and sensor inputs are similar to that of Alder, but it is programmed in C, rather than BASIC. Otherwise it is essentially the same as Alder.

In the next section we describe the Behaviour-based controller and self-organising scheme used to support the new location recognition scheme. We then describe the experimental setup and procedure adopted; following that, we present our experimental results. We finish with a brief discussion and conclusions.

[1] A much larger mobile robot called 'Ben Hope' is the second in the series and is still under construction.

2 Behaviour-based Control and Mapbuilding Process

Cairngorm's control structure consists of three independent behaviours. They are responsible for obstacle avoidance, wall seeking, and mapbuilding, respectively. There is no direct communication between these behaviours except through the world. In combination they lead to successful location recognition behaviour. The mechanisms used in each of these behaviours will now be described in a little more detail.

The *obstacle avoidance* behaviour is a preprogrammed and fixed behaviour that makes the robot turn left for as long as either or both of the two whisker sensors (one on the left and one on the right) are on, i.e., are in contact with an obstacle (the robot is designed to follow a wall on its right hand side). As soon as there are no signals from the whisker sensors the robot resumes moving forwards.

The *wall seeking* behaviour is also a preprogrammed and fixed behaviour. If the robot has not experienced a whisker contact for some preset period of time (about three seconds in our experiments) it makes a right turn (towards the wall). It continues to turn until a signal from the right whisker, or both the right and left whiskers, is received (typically it is just the right-hand-side one), upon which the obstacle avoidance behaviour introduces a left turn away from the wall. The robot then continues to move forward until either the set period of time has elapsed again or it makes contact with an obstacle. In this way, the robot is able to both avoid obstacles, follow walls, and negotiate the corners of its enclosure without these being explicitly specified tasks achieved using explicitly programmed strategies. In this sense they are examples of emergent functionality as defined by [Steels '91].

In the experiment presented here both the obstacle avoidance and wall following behaviours were preprogrammed. However, we have previously shown that our robots can acquire this kind of obstacle avoidance and wall following competences through learning from interaction with their environments (see [Nehmzow, Hallam & Smithers '89] for details).

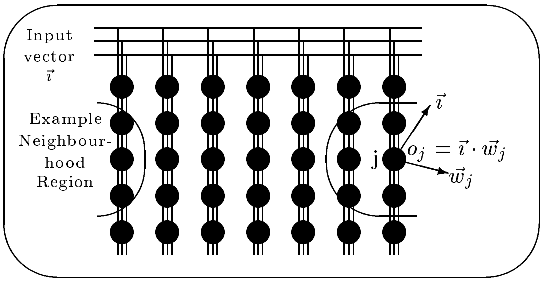

Figure 3: A two-dimensional self-organising network.

The third behaviour is *mapbuilding*. By 'mapbuilding' we mean something more like *taking notes* of particular experiences, rather than constructing a geographical map or floor-plan. For this a self-organising network (see [Kohonen '88]) is used. It is constructed ('trained') using input vectors derived from the motor action commands produced by the obstacle avoidance and wall-seeking behaviours. We next briefly review the details of this kind of network before describing in more detail the location recognition behaviour.

2.1 Self-Organising Networks

Consider the two-dimensional self-organising network (SON) given in figure 3.

Each cell j of the SON has an individual weight vector $\vec{w}_j$ of unit length. Each normalised input vector, $\vec{\imath}$, is fed to all cells[2]. The output o_j of cell j is determined by the dot product of input vector $\vec{\imath}$ and the weight vector $\vec{w}_j$ of cell j:

$$o_j = \vec{w}_j \cdot \vec{\imath} \qquad (1)$$

The cell with the strongest response, the largest output value, is selected. This cell as well as all neighbouring cells within a defined neighbourhood region are then modified according to the following equation:

[2] For self-organisation, normalising the input vector is not strictly necessary. However, for numerical comparison of two responses of the network, which we do, it is.

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

where η is the so-called 'gain'—a value that determines the amount of change (it is typically set at 0.2). Weight vectors outside the specified neighbourhood[3] remain unchanged. After several 'epochs', i.e. presentations of input vectors to the network, the net develops regions which respond most strongly to particular types of input vectors. In this way a mapping is developed whereby different input vectors are mapped onto different regions of the network (always an injection, often a bijection).

2.2 Location Recognition Behaviour

On Cairngorm we have used a one-dimensional, ring-shaped SON of 50 cells, as in earlier experiments with Alder, with a neighbourhood region of one node either side. As the robot makes its way around its enclosure, wall-following and avoiding obstacles, a moving average of durations of turn actions is computed. If a turn action occurs which takes longer than the current average (a 'significant turn action') an input vector is constructed and fed into the self-organising network and the modification to the weight vectors of the net as defined above is performed. After about two to three times round the enclosure the one-dimensional ring develops a stable enough structure to be used to recognise particular locations. This is demonstrated by instructing the robot (by pressing a microswitch mounted on it) to store the excitation pattern of the self-organising ring when the robot is at the location that is to be recognised ('home' location). This stored pattern is then compared with all subsequent excitation patterns. If the pattern at the current location of the robot is found to be sufficiently similar to the stored one, the robot indicates that it has arrived back at the home location. The comparison is computed using the 'city-block' distance ϵ between two excitation patterns, and is given by:

[3] The neighbourhood size is often chosen to be decreasing over time, so that self-organisation occurs over larger areas of the network early on and then becomes more local, but in our case we keep the neighbourhood region fixed.

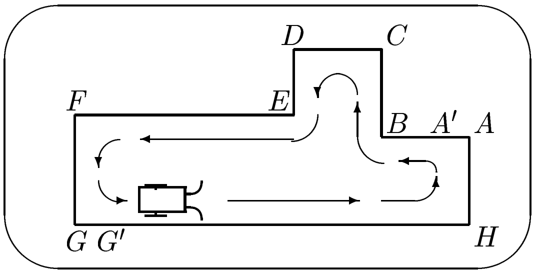

Figure 4: The experimental enclosure for location recognition.

$$\epsilon = \sum_{k=1}^{50} |o_{s_k} - o_{c_k}|, \qquad (3)$$

where $\vec{o}_s$ is the stored excitation pattern, and $\vec{o}_c$ is the current excitation pattern. By using the city-block distance measure (or Euclidean distance measure, for that matter) we are effectively performing a vector quantisation, so the property of topology-preserving mapping of SON is not exploited. The advantage of this is that the network can be used for location recognition even in the early learning stages. However, once the net is well settled the index information of the most excited cell alone should suffice to recognize locations.

3 The Experimental Results

In this section we present our experimental results. We begin by describing the experimental setup.

3.1 Experimental Setup

The robot is placed in a simple enclosure, containing rightangled convex and concave corners as well as straight walls. Figure 4 shows the layout of the enclosure. The letters indicate locations where a turn action usually exceeds the average duration and therefore where an input vector to the SON is usually generated.

Once started, the robot follows the wall on its right hand side round the enclosure. This is

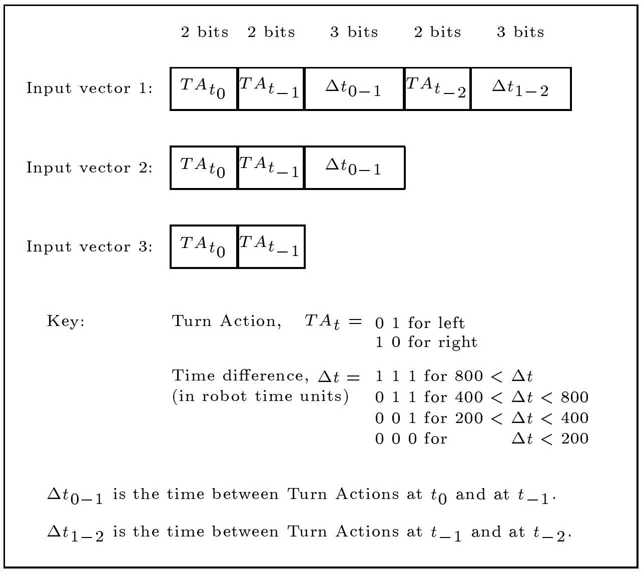

Figure 5: Input vector definition for the Self-Organising Controller.

achieved, as we mentioned before, by a combination of the obstacle avoidance and wall seeking behaviours. The mapbuilding behaviour monitors the turn actions commanded by the other two behaviours. Whenever a turn action takes longer than the current average turn action time, an input vector (see input vector 1, in figure 5), is generated and presented to the SON. The response of each cell in the ring is then determined and the relevant weight vectors are updated, as described in section 2.1. The robot is left to wall follow its way around the enclosure for typically two to three rounds, then a "home" corner is selected by pressing the microswitch mounted on the robot. This results in the current excitation pattern being stored in memory. From this point on all subsequent excitation patterns of the SON are compared to the stored pattern. If the city block distance between the goal-pattern and the current excitation patterns is less than $\frac{1}{3}$ of the average city-block distance, the robot assumes it is back at the goal location and a red light (mounted on the interface board) lights up.

As can be seen from input vector 1 in figure 5, the information used as input to the self-organising network consists of the type of the current turn action (right or left), and the types of the previous two turn actions, together with the elapsed time between each pair.

In order to investigate the performance of this location recognition scheme we used two other input vectors, see input vector 2 and 3 in figure 5, each one containing less information than the previous one. Input vector 2 consists of the types of the current and previous turn actions and the elapsed time between them, and input vector 3 consists of just the types of the current and previous turn actions. The experimental setup when using each of these three kinds of input vector was in all other respects the same.

The results obtained for all three input vectors are presented in the next section.

3.2 Experimental Results

Table 1 shows the location recognition results obtained using input vector 1 (see figure 5).

Location	Recognitions	Total no. of Visits
H	3	4
B	3	3
C	3	3
D	3	3
E	3	3
F	3	3
G	3	3
G'	2	4
A	3	3
A'	3	3

Table 1: Location recognition using input vector 1.

The letters refer to the locations shown in figure 4. Except for A' and G', they correspond to the corners of the enclosure. This is because turn actions that take longer than average are produced at these places. The locations A' and G' are produced because the robot typically takes two long turn actions to negotiate these particular corners.

The one 'missed' recognition at corner H occurred early on in the run and was due to the network not having settled down enough for a successful recognition to be registered. The two 'misses' at G' occurred because on these occasions the robot got round corner G with one turn action, and so the region of the SON corresponding to an input vector having G', G, and F was not excited by any input vector.

Table 2 indicates the pairs of locations which were confused during the run.

	A	A'	B	C	D	E	F	G	G'	H
A										
A'									X	
B										
C										
D										
E										
F										
G										
G'		X								
H										

Table 2: Confused locations when using input vector 1.

In this case it was only locations A' and G' which were confused. This occurs because the input vectors at these locations (built from H, A, A' and F, G, G' respectively) are similar, and thus excite the self-organising network in the same way.

Table 3 presents the location recognition results when using input vector 2. As can be seen, there is a degradation in performance with 'missed' locations occurring at D, F, and G. The failure to recognise location F here was due to an input vector not being generated at location E, thus leading to an 'odd' vector at F. This occurred only once and was the result of the inevitable variation in actual behaviour experienced when using real robots (even simple ones) in a real environment. This was, however, a one-off event and never observed to occur again.

Location	Recognitions	Total no. of Visits
H	3	3
B	3	3
C	3	3
D	3	4
E	3	3
F	3	4
G	4	5
G'	3	3
A	3	3
A'	3	3

Table 3: Location recognition using input vector 2.

	A	A'	B	C	D	E	F	G	G'	H
A		X			(X)			X	X	
A'	X							X	X	
B										
C										
D										
E										
F										
G	X	X			(X)				X	
G'	X	X						X		
H										

Table 4: Confused locations when using input vector 2.

The degradation in performance can be seen more clearly in table 4, which presents the pairs of confused locations. The bracketed pairs denote occasional confusions.

When we reduce the amount of information in the input vector still further, and use input vector 3, we obtain the results presented in tables 5 and 6. Here we can see that the robot is still able to recognise locations, but its ability to distinguish certain pairs of them is significantly diminished.

Location	Recognitions	Total no. of Visits
B	3	3
C	3	3
D	2	2
E	2	2
F	2	2
G	2	2

Table 5: Location recognition using input vector 3.

	A	A'	B	C	D	E	F	G	G'	H
A		X			X			X	X	X
A'	X				X			X	X	X
B						X				
C							X			
D	X	X						X	X	X
E			X							
F				X						
G	X	X							X	X
G'	X	X						X		X
H	X	X						X	X	

Table 6: Confused locations when using input vector 3.

As we would expect, the performance of the whole system depends on the information put into the self-orgainsing network. If the input vector contains insufficient information, then reliable location recognition is impossible. From the results presented above we can see that reducing the information content of the input vector does not affect the robot's ability to recognise a non-wall type environmental feature (corners in this case), but it does affect its ability to differentiate some pairs of such locations. The less informative the input vector, the more locations "look the same" to the robot.

4 Discussion

4.1 Discovering Significant Environment Structure

In our previous location recognition scheme based upon motor actions we required a set of seven self-organising two-dimensional networks to get the robot to reliably recognise (and distinguish between) locations. Yet in the scheme we presented above we only need one one-dimensional network (ring) to achieve similar performance results. We can explain this effect by observing that our new scheme works by filtering the sequence of motor commands so that only those *not* generated by following a straight wall are used to build input vectors to the self-organising structure, whereas in the previous scheme the set of seven networks had to do this filtering work implicitly.

Another way of viewing this is to say that the sequence of motor commands produced as the robot wall follows its way around the enclosure contains two kinds of structure. One kind occurs at a high frequency and is produced by the wall-following actions. The other kind has a lower frequency and is produced by the corners, or non-straight-wall features of the enclosure. It is this second type of structure in the motor action commands that contains information about significant structure in the robot's environment (corners in this case); the first type merely reflects the fact that the robot's environment has straight walls in it—a rather less useful piece of information for location recognition.

4.2 Setting Thresholds

When we first started to experiment with the scheme presented above, the turn action time threshold, used to distinguish wall-following actions from other actions, was set by hand. We arrived at the particular threshold value by carefully observing the robot's behaviour and choosing it such that it would differentiate between motor actions performed at a 'significant' location (usually a corner) and those performed elsewhere (while wall following). Later we implemented the moving average calculation, thus removing the need for this 'magic number' to be set by hand (by doing this we have effectively introduced two other values to be set: the time window over which the average duration of motor actions is computed and the proportion by which the duration of a motor action has to exceed this average to become a "significant action". However, both these values are not very critical and far easier to set than the previously needed thresholds). Similar mechanisms are found in biology. Pigeons, for example, extract the changes in air pressure generated by changes in altitude of a few feet by ignoring the total strength and only measuring differences around some mean, [Gould '82]. Using this simple device in our robot means that it does not need any predefined knowledge about thresholds and significant motor actions, it finds this out for itself. This approach has further advantages: The robot is able to adjust its assessment if the world or the robot change. Cairngorm's moves, for example, become slower with decreasing battery charge. If a fixed threshold for determining significant moves was used, the performance of the robot would change, simply because motor actions take longer. However, because the average turn action will also take longer, Cairngorm is able to adjust for this and so maintain its performance.

A similar threshold value is used in the comparison between the stored ('home') excitation pattern and all subsequent patterns. Once again, we started by determining the required value for this empirically and setting the value in the program by hand. But having devised the successful averaging mechanisms to set the turn action time threshold we decided to try a similar device for the comparison threshold. The robot thus computes a moving average of all the city-block distances between the stored and current excitation patterns. Once a distance is smaller than $\frac{1}{3}$ of this average distance, the robot indicates that it recognises the current pattern as being the same as the stored pattern, and thus that it recognises its current location as the 'home' location.

4.3 Why 'Motor Actions' and not 'Sensor Signals'?

Sensing and acting are so tightly coupled that it seems fair to say that they are two sides of the same coin. In earlier experiments we have used schemes based on sensory information to recognise locations (see section 1). In the experiments reported here we have obtained the same results as in the earlier work, using motor information rather than sensor information. This was done to prove exactly this point: that sensing and acting are tightly coupled.

There can be practical advantages in using motor action information: the amount of information to be processed is often smaller because the robot is effectively acting as an analog computing device. Also (and perhaps more importantly for practical applications), the controller becomes independent of the actual sensors used. If necessary, sensors can be replaced without affecting the actual controller.

4.4 Comparison with Animal Navigation

There are numbers of well studied examples of navigation by learned location recognition in the biological literature. In an exploratory phase bees store *visual images*, which are then associated with a motion vector that gives the direction towards the hive, see ([Waterman '89], [Gould & Gould '88] and [Cartwright & Collett '83]). When released some distance away from the hive, bees can find their way back using these acquired visual images by comparing them with the current images and calculating the appropriate flight direction to take them to their hive, [Collett '87]. In a similar way our robot stores "motor action images" and later compares stored images with current ones in order to recognise locations it has been at before. However, no vectors pointing homewards are associated with these images.

Other kinds of sensor-based navigation behaviour schemes have also been documented. For example, pigeons and salmon navigate using *olfactory information* ([Gould '82], [Hasler, Scholz & Horral '78]), and it seems possible that pigeons also use *acoustic (infrasound) and magnetic sensors* to find their way home [Gould '82]. Bees and ants also use the *polarisation of the light* of the sky to navigate [Wehner '76].

All these biological examples appear to just use sensor information and they are clearly reliable and robust. However, our experiments with simple robots have shown that motor-action based schemes can be used as well in order to achieve high degrees of reliability and robustness. The reasons for this apparent difference are not clear to us.

5 Summary and Future Work

In earlier work we demonstrated location recognition schemes using sensor-based feature detectors and using motor action changes. In this paper we present a new location recognition scheme, which again uses information about motor actions, but which is simpler and yet equally reliable and robust. As a part of this new scheme we have also incorporated automatic adaptive threshold setting mechanisms to reduce the number of parameters ('magic numbers') which have to be set by hand.

In our ongoing series of experiments to investigate the autonomous acquisition of specified competences we have demonstrated in separate schemes the acquisition of obstacle avoidance, wall-following, corridor following and location recognition competence by the robot. Our plan for the future is to put all these schemes together in one robot to demonstrate the staged learning of all the competences required for simple navigation in an autonomous robot.

Acknowledgements

The work reported here is supported by a grant from the UK Science and Engineering Research Council (grant number GR/F/5852.3). Other facilities were provided by the Department of Artificial Intelligence at Edinburgh University. We would like to thank John Hallam for the stimulating discussion about this work. We thank Claudia Alsdorf, Peter Forster and Barbara Webb for reading earlier versions of this paper and their helpful comments. Thanks also to the staff of the mechanical, electronic, and photographic workshops of the Department of Artificial Intelligence for their help and support.

References

[Cartwright & Collett '83] B.A. Cartwright and T.S.Collett, *Landmark Learning in Bees*, Journal of Comparative Physiology (1983) 151:521-543.

[Collett '87] T.S.Collett, *Insect Maps*, TINS, Vol. 10 No. 4, 1987.

[Gould '82] James L. Gould, *Ethology: The Mechanisms and Evolution of Behavior*, W.W. Norton and Co., New York 1982.

[Gould & Gould '88] James L. Gould and Carol Grant Gould, *The Honey Bee*, Scientific American Library 1988, p. 106.

[Hasler, Scholz & Horral '78] Arthur D. Hasler, A.T. Scholz and R.M. Horrall, *Olfactory Imprinting and Homing in Salmon*, American Scientist 66 (1978) pp. 347 - 55; quoted in [Gould '82].

[Kohonen '88]
Teuvo Kohonen, *Self-Organization and Associative Memory*, Springer Verlag, Berlin, Heidelberg, New York, 1988.

[Nehmzow, Hallam & Smithers '89]
Ulrich Nehmzow, John Hallam and Tim Smithers, *Really Useful Robots*, in: T. Kanade, F.C.A. Groen and L.O. Hertzberger (eds.), *Intelligent Autonomous Systems 2*, pp. 284–293, ISBN 90-800410-1-7, Amsterdam 1989.

[Nehmzow & Smithers '91]
Ulrich Nehmzow and Tim Smithers, *Mapbuilding using Self-Organising Networks*, in: Jean Arcady Meyer and Stewart Wilson (eds.), *From Animals to Animats*, pp. 152–159, MIT Press Cambridge Mass. and London, England, 1991.

[Nehmzow, Smithers & Hallam '91]
Ulrich Nehmzow, Tim Smithers and John Hallam, *Location recognition in a mobile robot using self-organising feature maps*, in: Günther Schmidt (ed.), *Information Processing in Autonomous Mobile Robots*, Springer Verlag, Berlin, Heidelberg, New York, 1991.

[Steels '91] Luc Steels, *Towards a theory of emergent functionality*, in Jean Arcady Meyer and Stewart Wilson (eds.), *From Animals to Animats*, pp. 451–461, MIT Press Cambridge Mass. and London, England, 1991.

[Waterman '89] Talbot H. Waterman, *Animal Navigation*, Scientific American Library 1989, p. 183.

[Wehner '76] Rüdiger Wehner, *Polarized-Light Navigation by Insects*, Scientific American 235 No. 1 (1976), pp.106-15.

[Willshaw & v.d.Malsburg '76] David J. Willshaw and Christoph von der Malsburg, *How patterned neural connections can be set up by self-organization*, Proc. R.Soc. London B. 194, pp.431-445 (1976).