Middle Scale Robot Navigation - A Case Study

Carl Owen and Ulrich Nehmzow
Department of Computer Science
University of Manchester
Manchester M13 9PL
United Kingdom
owenc@cs.man.ac.uk
u.nehmzow@cs.man.ac.uk

7/4/97

Abstract

In this paper we present results of experiments carried out with a route learning system for a mobile robot, conducted in a 'real world' environment covering distances of several hundred metres. The system uses no odometry and is based on a self-organising mapbuilding process using perceptual landmarks.

A performance metric is defined and used to measure the robot's ability to traverse the route.

1 Introduction

Much of the work to date in the field of mobile robot navigation has been conducted in small scale laboratory environments - the question arises of how these systems will scale up when faced with more complex environments covering much larger distances.

In the experiments reported here a route learning system for mobile robots is tested in a 'real world' environment covering several hundred metres. We call this "middle scale navigation", following the definition from the biological sciences for navigation that typically encompasses foraging trips of insects covering journeys of up to $10^6\times$ body length [12].

The route learning system uses no metric information whatsoever, therefore obviating the need to compensate for drift error, and no pre-installed world model is represented. The robot is trained to follow a route, and a performance metric is introduced with which to measure the robot's increasing ability to traverse the route with each successive training session.

The paper is structured as follows: Section 2 details the mapbuilding and route learning mechanism. Section 3 describes the route on which the robot was trained. A description of the experimental method used is given in section 4 along with the results obtained in the experiment, whilst section 5 details the environmental conditions under which the experiment was conducted. A discussion of perceptual aliasing is given in section 6, followed by conclusions/further work in section 7.

2 The mapbuilding and route learning mechanism

The robot used in the experiments is a Nomad 200 mobile robot (see figure 1). This robot is equipped with 16 sonar sensors, 16 infra-red sensors, 20 tactile sensors and a monochrome CCD camera; in these experiments only the sonar and infra-red sensors were used.

The route learning system was developed at Manchester and uses a Self-Organising Feature Map (SOFM) [4] to build a representation of the environment (using the robot's sensory perception) as the robot is guided along the route by an operator.

Proc. AISB workshop on "Spatial Reasoning in Mobile Robots and Animals", Manchester 1997.
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 - 6161.
Report number UMCS-97-4-1. http://www.cs.man.ac.uk/csonly/cstechrep/titles97.html

Figure 1: The Nomad 200 mobile robot

The SOFM is an Artificial Neural Network (ANN) that performs a topological clustering of its input data using an unsupervised learning mechanism. The network consists of one layer of cells typically arranged as a two dimensional grid. Figure 2 shows the basic structure of such a network.

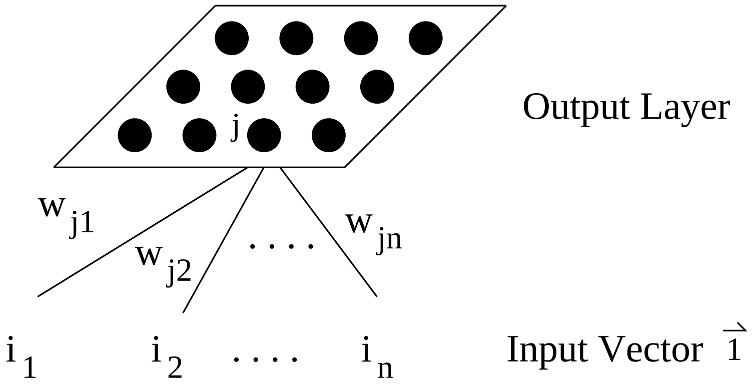

Figure 2: Structure of the SOFM

Each unit of the network receives the same input pattern (in this case consisting of the sonar and infra-red readings of the robot). This pattern is presented to the network as a vector:

$\vec{\imath} = [i_1, i_2, ..., i_n]$

These inputs are generated as the robot is led along the route. The weights on the connections to a single output unit are given by:

$\vec{w_j} = [w_{j1}, w_{j2}, ..., w_{jn}]$

where j identifies unit j within the output layer and n is the nth element of the input vector. These weight vectors are normalised to constant length (see [4]).

To find the output o_j of unit j we calculate the weighted sum of its inputs:

$$\sum_{k=1}^{n} w_{jk} i_k = \vec{w_j} \cdot \vec{\imath} \tag{1}$$

with n being the number of elements in the input vector.

In its initial state each cell of the network has a unique set of weights. Therefore, when presented with an input vector, one cell will respond more strongly than the others.

The weight vector of this "winning" unit along with its eight neighbours is updated according to equation 2:

$$\Delta w_{jk} = \alpha(i_k - w_{jk})$$

and

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk} \tag{2}$$

where α is the learning rate parameter. Typical values for this parameter are in the range 0.2 - 0.5. Weight vectors are normalised again after being adjusted.

As this process continues the network organises into a state whereby dissimilar input vectors/patterns map onto different regions of the network, whilst similar patterns are clustered together in groups. In using the SOFM to process the robot's sensor signals, distinct physical locations (with distinct 'perceptual signatures' - the sensory patterns represented in the input vectors) will map onto distinct regions of the network, with similar perceptual patterns clustering together in a region. In this way, regions of the network can be seen as representing 'perceptual landmarks' within the robot's environment.

Used in this manner the SOFM provides a mapbuilding mechanism that is suitable for localisation tasks in mobile robots (see also [3, 6, 7, 9]). For route learning the SOFM can be used to associate locus and action by extending the input and weight vectors to include the required actions. During route training, actions form part of the input vector and thus become part of the weight vector of the trained units[1].

On route recall the action part of the input vector is set to zero, and thus the winner is calculated on sensor data alone; due to the topological mapping achieved by the SOFM this winner is likely to be within the same region that was excited at this physical location during training. The action associated with this location can then be retrieved directly from the winner's weights. The result is route following based on landmark identification, using no metric information whatsoever.

Figure 3 shows the principle of the SOFM as associator (see also [3]).

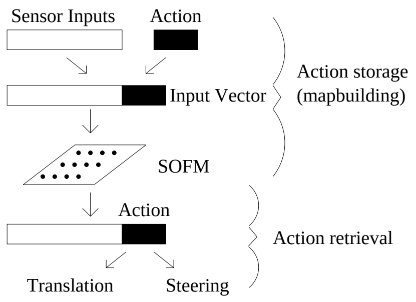

Figure 3: SOFM as associator

Figure 4 shows the sequence of events for action retrieval. Firstly the robot receives its sensory information (the dotted lines in the diagram indicate infra-red or sonar signals being projected from the robot). The input vector is now formed by inserting zeros into the action part of the vector, and this is then fed into the network and a winner found. Finally, the action is retrieved from the action section of the winner's weights (a more detailed description of this system can be found in [10]).

3 The route

The route from which the results were taken is shown in figure 5.

This route is along the corridors of the first floor of the computer building at Manchester university, these corridors present a busy thoroughfare with access to both the adjoining IT building and the departmental library.

[1]The action required at a location is read directly from the joystick as the robot is guided along the route.

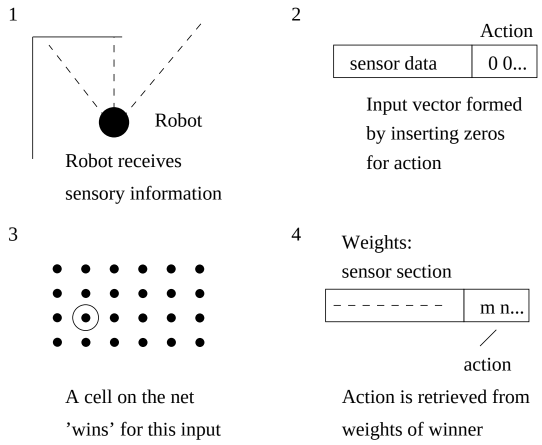

Figure 4: Action retrieval sequence

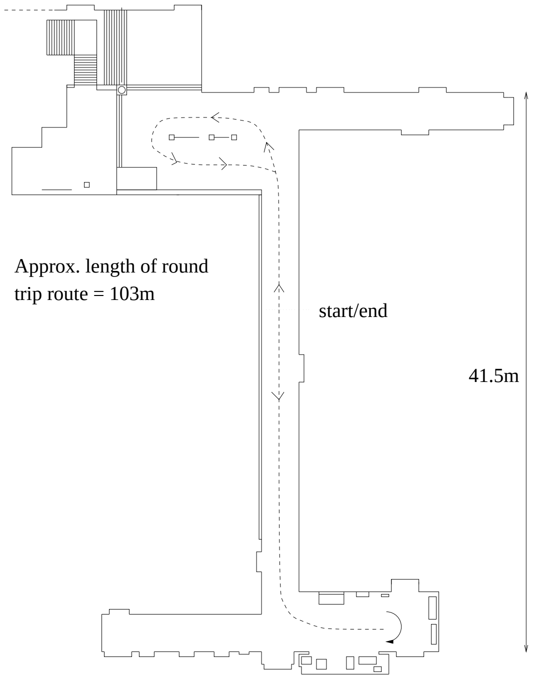

Figure 5: The route

3.1 Odometry drift

Figure 6 shows a robot trace of the route using the robot's onboard odometry mechanism.

Since the system described here makes no use of metric information, there is no need to compensate for the accumulated error (for examples of correction methods see e.g. Kalman filter [5], evidence grids [13]).

4 Method

In our experiments the robot was trained in stages, for each stage the robot is led around the route by the operator for one complete circuit - this represents one training phase or round. At the end of each training round the robot is 'let loose' at the start point to traverse the route autonomously. The distance taken between each 'failure' point (as defined below) along the route is then recorded and the

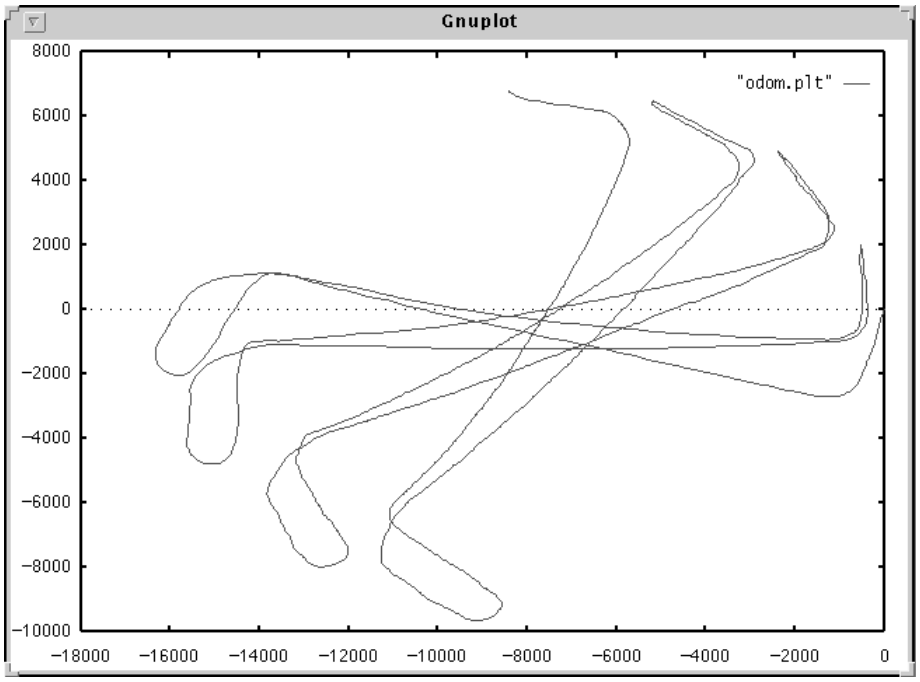

Figure 6: Odometry drift (over 4 circuits of route)

Mean Distance Between Failures (MDBF) calculated according to equation 3:

$$\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} DBF_i, \tag{3}$$

with n being the number of readings taken, and DBF_i being the ith 'Distance Between Failure' reading. This metric gives us an indication of the robot's increasing (MDBF increasing) or decreasing (MDBF decreasing) ability to traverse the given route with each training round.

The definition of 'failure' in the robot's route following task is as follows:

- The robot touches an object (or wall).

  or

- $\|$ required heading - observed heading $\| > 90°$

Once a failure has occurred the robot is returned to a position on the route just before the point of failure. The robot is then guided further along the route until it is able to pick up the route by itself, at which point the next reading commences (see figure 7).

In the experiments reported here, we collected 35 distances between failure (n = 35) before training the robot once again over a complete round. This figure was chosen as a compromise between the requirements for an expressive measurement and a reasonable timescale for collection of the data (bearing in mind the battery life of the robot). Once the robot was able to complete 15 rounds without a single failure it was deemed to have successfully learned the route and the experiment was terminated[2].

To achieve this, five training rounds were necessary in the environment given in figure 5. Figure 8 shows the MDBF in metres for each training round (the 'dot' on the graph); as can be observed from the graph the robot does indeed improve in its route following capability with each successive training round.

[2]Since one complete round of autonomous route following takes approximately 16 minutes, 15 rounds approximates to 4 hours running time - this is approaching the maximum running time for the robot on one set of batteries.

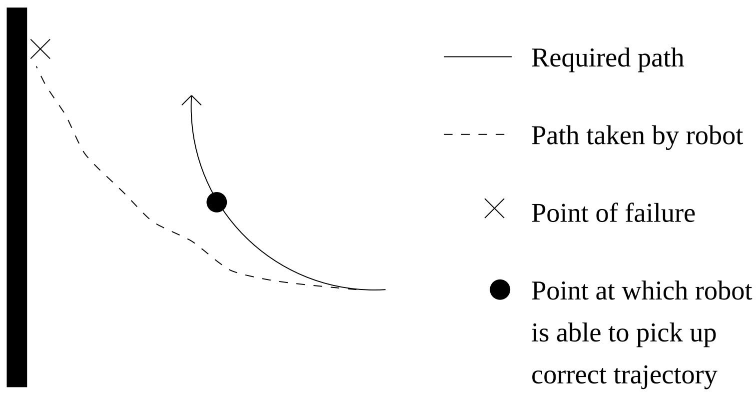

Figure 7: Start of the next failure reading

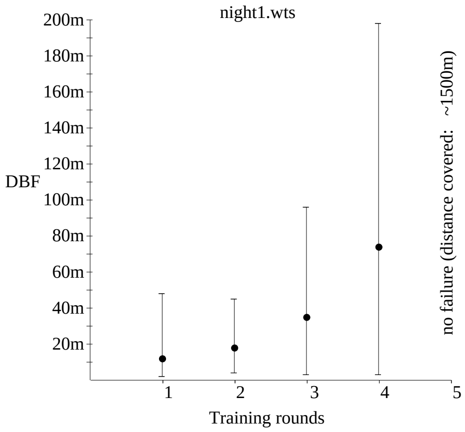

Figure 8: MDBF results (the dot indicates the mean, the bars indicate minimum and maximum readings)

5 Environmental conditions

Training of the robot was carried out in the evening between the hours of 6pm and 3am. Table 1 shows the number of people moving past the robot during the training and recall phases of the network training session detailed in section 4.

Training round	no. people passing	
	Training phase	Recall phase
1	3	3
2	2	6
3	2	5
4	4	4
5	2	9
Average	2.6	5.4

Table 1: Summary of environmental conditions

5.1 Daytime testing

A further test was carried out with the fully trained network during the day when the corridors were much busier. In these conditions the robot was unable to complete 15 consecutive rounds without fail and an MDBF of 66.25 m was recorded.

During the course of taking these measurements it was noted that failure was caused by people sitting or standing within the environment rather than simply walking past the robot. This phenomenon can be explained by the fact that objects in motion cause only temporary fluctuations to the environment and, if they pass quickly enough, the robot is able to re-gain the 'correct' perceptual signature for its current location. Stationary people, on the other hand, may be identified as landmarks and cause confusion to the route follower. If, however, the change to the perceptual signature is small, then the generalisation properties of the network ensure that the same node or one of its neighbours will 'fire', generating 'correct' behaviour despite the disturbance.

6 Perceptual aliasing

The size of the network used in these experiments was 1600 cells (a 2 dimensional grid of 40×40). The average number of cells involved in traversing one circuit of the route with the fully trained network was 258, this average was taken over five consecutive circuits of the route[3]. Of these cells an average of 28 were involved in perceptual aliasing, which represents approximately 11% of the total involved in route traversal (we define a cell as being involved in perceptual aliasing if it is repeated at a location, or locations, other than the one at which it was first encountered).

Roughly following Ballard and Whitehead's definitions [1] we can define two types of perceptual aliasing in the context of our route following task:

1. Propitious aliasing - $(signatureA = signatureB) \wedge (actionA = actionB)$
2. Destructive aliasing - $(signatureA = signatureB) \wedge (actionA \neq actionB)$

Since the perceptual aliasing measured in our experiments does not interfere with the robot's route following ability we can classify it as type 1. This type of perceptual aliasing is obviously desirable in terms of network storage capacity. An effect similar to destructive aliasing can be observed during the early stages of training; here locations with signatures that are similar in some way but with different actions can get confused until the network has had sufficient 'experience' at the locations to pick up the differences between the signatures.

An example of this early interference is illustrated in figure 9. It was observed that, after the second round of training, some of the locations at which the robot had performed the correct action for the first round were now failure points. In the figure location 'A' is an example of a location at which this phenomenon occurred.

On inspection of the network it was found that the winning nodes for locations 'A' and 'B' were within the same neighbourhood (i.e. the network had 'identified' them as the same landmark). The effect of this was that training given at location 'B' on the second round, there being none required at 'A', had altered the trajectory at 'A' sufficiently to cause failure (in effect the nodes in this neighbourhood have received twice as much training at 'B' as at 'A'). By the end of the third training round the winning nodes for these locations were not contained within the same neighbourhood, i.e. destructive aliasing no longer occured.

7 Conclusions and future work

In this paper we have given results of tests carried out with a route learning system over a middle scale distance (a mobile robot was trained to follow a route of just over 100m in length and was able to follow it without error 15 times, at which point the experiment was terminated). The system uses perceptual landmarks and a mapbuilding process based on self-organisation. Odometry is not used.

A metric (Mean Distance Between Failures) was introduced and used to measure the robot's route following performance; the results show a clear increase in navigational competence after each subsequent training session. We maintain that quantitative measures of robot behaviour such as this are required to determine the influence of system components upon the overall behaviour of a robot control system,

[3]The figure of 1600 cells was chosen to ensure a network of sufficient size for the given environment, as can be seen from the results this network was far too large (see section 7 for discussion of this issue).

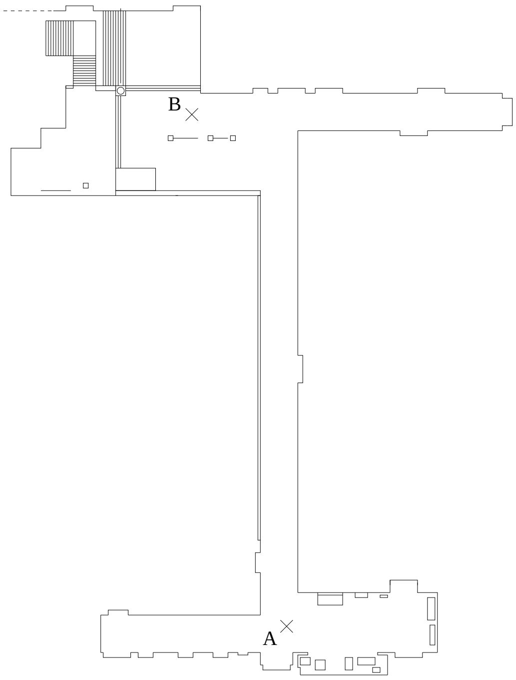

Figure 9: Network interference

as well as to facilitate comparison between different control systems, different robot platforms, different environments, e.t.c. - in other words, they are required tools for the science of robotics (see also [11]).

The route learning system was shown to cope with temporary changes to the environment (moving people), however 'permanent' changes (stationary people) could cause failure. As the map developed by the robot can be used for prediction of future perceptions, one way to solve this problem would be to detect deviations from expected perception and to use this information to decide an appropriate action. This approach is currently under investigation.

One must assume that on longer and/or more complex routes 'destructive aliasing' would become more prevalent. One approach to solving this problem would be to use a 'history' of past perceptions (or alternatively past nodes) as part of the input vector to disambiguate locations (see e.g. [7, 9]). However, one of the problems with this approach is that it would also have a negative effect on propitious aliasing. An alternative might be to carry a 'floating history', with the nodes containing a list of possible actions rather than one action directly associated with a sensory pattern. The operator, on observing destructive aliasing, could 'mark' the offending location (i.e. node) with an alternative action; on recall the appropriate action for a marked node would be selected according to the current history (history 'as and when required').

Network capacity also needs to be considered. In these experiments the network size chosen was far larger than required for the given environment (see section 6). However, it is not possible to determine, by simply observing the environment, what size of network would be optimal. Rather than using a fixed size of map, a better approach would be for the robot to autonomously determine the approximate size of map required according to the complexity of its surroundings. A rough guide could perhaps be achieved by counting the number of 'different' input vectors gathered on one circuit of the route (difference being defined by a pre-determined similarity measure). An alternative approach might be to use growing networks such as Fritzke's "neural gas" network [2].

The system as it stands is restricted to following a canonical path (the information contained in the map is similar to 'route knowledge', see [8]), and as such is unsuitable for free navigation. However, the perceptual clustering mechanism could be augmented with relational information such as distance and direction to other locations. Since this metric information is only local (i.e. between locations), correctional methods such as those mentioned in section 3.1 are not required. A system of this type is currently under construction.

References

[1] D. H. Ballard and S. D. Whitehead. Learning visual behaviours. In H. Wechsler, editor, *Neural Networks for Perception*, volume 2, pages 8–39. Academic Press, 1992.

[2] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge Mass., 1995.

[3] J. Heikkonen. *Subsymbolic Representations, Self-Organising Maps, and Object Motion Learning*. PhD thesis, Lappeenranta University of Technology, Finland, 1994.

[4] T. Kohonen. *Self Organisation and Associative Memory*. Springer Verlag, Berlin, Heidelberg, 1988.

[5] A. Kurtz. Constructing maps for mobile robot navigation based on ultrasonic range data. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 26(2):233–242, April 1996.

[6] U. Nehmzow and T. Smithers. Mapbuilding using self-organising networks. In J. Arcady Meyer and S. Wilson, editors, *From Animals to Animats*, Proceedings of SAB90 Paris, pages 152–159. MIT Press, Cambridge Mass. and London, England, 1991.

[7] U. Nehmzow and T. Smithers. Using motor actions for location recognition. In F. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems*. MIT Press, 1992.

[8] J. O'Keefe. Computations the hippocampus might perform. In L. Nadel, L. Cooper, P. Culicover, and R. M. Harnish, editors, *Neural Connections, Mental Computation*. MIT Press, Cambridge Mass., 1990.

[9] C. Owen. Landmarks, topological maps and robot navigation. Master's thesis, Manchester University, England, 1995.

[10] C. Owen and U. Nehmzow. Route learning in mobile robots through self-organisation. In *Euromicro workshop on advanced mobile robots*. IEEE Computer Society, ISBN 0-8186-7695-7, 1996.

[11] T. Smithers. On quantitative performance measures of robot behaviour. *Robotics and Autonomous Systems*, 15(1–2):107–133, 1995.

[12] R. Wehner, B. Michel, and P. Antonsen. Visual navigation in insects: coupling egocentric and geocentric information. *J. Exp. Biology 199*, pages 129–140, 1996.

[13] B. Yamuchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 26(3):496–505, June 1996.