[0]Carl Owen and Ulrich Nehmzow, Route Learning in Mobile Robots through Self-Organisation.
Published by IEEE Computer Society, Proc. Eurobot 96. Kaiserslautern 1996.

Route Learning in Mobile Robots through Self-Organisation

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

Abstract

This paper describes route-learning experiments with an autonomous mobile robot in which mapbuilding is achieved through a process of unsupervised clustering of sensory data. The resulting topological mapping of the robot's perceptual space is used for subsequent navigation tasks such as route following.

After the autonomous mapbuilding process is completed, the acquired generalised perceptions are associated with motor actions, enabling the robot to follow routes autonomously. The navigation system has been tested extensively on a Nomad 200 mobile robot, it is reliable and copes with noise and variation inherent in the environment.

One important aspect of the mapbuilding and route following system described here is that relevance or irrelevance of perceptual features is determined autonomously by the robot, not through predefinition by the designer. Secondly, further to previous work ([6]) the presented route learning system enables the robot to use the map for association of perception with action, rather than localisation alone.

1. Introduction

Until recently most approaches to robot navigation were based on the use of symbolic representations of the environment to determine the actions required for the robot to reach its goal location. These approaches typically use computational geometry techniques for this task (see e.g. [1, 3]).

This use of symbolic representations relies heavily on a sufficiently detailed and accurate model of the environment, either pre-defined or built by the robot through exploration. However, supplying a pre-defined model that is accurate enough for reliable navigation on anything but the simplest of routes is not always possible, and the cost of building such a model can be high in terms of both time and memory required. Inaccuracies in input information can also cause problems for such model-based navigation systems, even small perturbations (due to sensory noise for example) can seriously degrade performance and, at worst, cause the system to lose its bearings completely and be unable to complete its task.

One of the more fundamental problems with the symbolic paradigm is the split that exists between the high level symbolic processing on the one hand, and the low level (subsymbolic) processing that deals with the information gathered via the sensory devices on the other. At some point in the process a match has to be made between the information received by the systems sensors, and some internal symbolic model. In machine vision for example, the internal models might include objects or features the system could expect to find within its working environment (see e.g. [10]). As a consequence, much intermediate processing is required to bridge the gap between sensory information and symbolic object identification (see e.g. [5]), and for the system to work in real time (as it must) this processing must be done very quickly.

So the issues we need to address in order to build a successful robot navigation system are:

- What is a suitable mapping mechanism for the task of autonomous navigation?
- How should such a map be used to generate the required actions on the robot?
- What mechanisms can we use to enable the robot to cope with the inevitable noise and variation of the real world?
- Since the real world is so complex to model, simulation is not adequate (if at all possible - see [8]). Testing of the system must be done in the real world, using

situated agents that interact directly with the environment.

We must also bear in mind that the mechanisms we use must be computationally cheap, in order to be usable in real time.

In this paper we present a robot navigation system that enables a mobile robot to build its own internal representation of its environment and use it for autonomous navigation. With this system the robot is trained, rather than programmed, to follow an arbitrary route. The representation of the robot's environment emerges through self-organisation as the robot is guided along the route, and represents a topological, rather than geometrical, mapping of the environment. The map is built using low level (subsymbolic) information only, requiring no pre-defined symbolic models for its construction.

The paper is structured as follows: Section 2 deals with the map building and navigation mechanism. It outlines the advantages of using a Self-Organising Feature Map (SOFM) [4] to build the robot's representation of its environment, and discusses the implementation details of the system used in the experiments presented in this paper. A description of the training method used is also given. Section 3 sets out the results obtained in several experiments conducted in and around the robotics laboratory of Manchester University.

2. The mapbuilding and navigation mechanism

At the heart of the mapbuilding and navigation system is a Self-Organising Feature Map (SOFM) [4].

The SOFM is an example of 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 1 shows the basic structure of such a network.

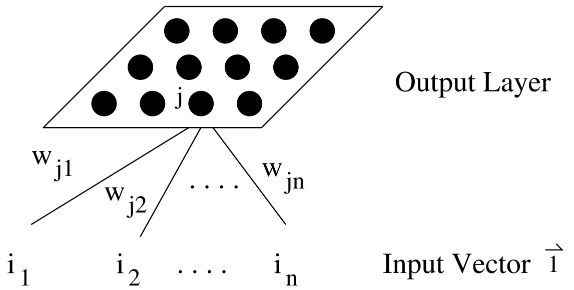

Figure 1. Structure of the SOFM

The input pattern containing sensory information (see section 2.2.1) is represented as a vector:

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

This input vector is presented to each output unit of the network, 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]).

The output o_j of unit j is found by calculating the weighted sum of its inputs, given by:

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

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

The initial state of the network has randomised values for the weights. Therefore one cell of the network will respond more strongly than the others to a particular input vector.

The weight vector of this "winning" unit as well as those of the eight neighbouring units are then changed 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.

Processing sensor signals using a SOFM generates a representation of the sensory information at the subsymbolic level, with less need for predefinition by the user than model-based or cartesian navigation systems.

When the network has settled, 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. To achieve this, no symbolic representations have been created, and the robot is mapping its environment "as it sees it".

In this way, regions of the network can be seen as representing 'perceptual landmarks' within the robot's environment, and from this point navigation can be achieved by the association of actions with landmarks. This provides an alternative to the geometric maps discussed in section 1, with the representation here emerging through self-organisation requiring no interpretation of the input data for its construction.

2.1. Experimental setup

The robot used for these experiments was a Nomad 200 mobile robot (see figure 2), named (after its serial number) “FortyTwo”.

Figure 2. The Nomad 200 mobile robot “FortyTwo”

This hexagonal robot is equipped with sixteen ultrasonic range finding sensors (range up to 6.5 m), sixteen infrared (IR) sensors (range up to 60cm), twenty tactile sensors and a monochrome CCD camera.

The robot is driven by three independent AC motors for translation, steering and turret rotation. A 486 PC is the main controller with several slave processors (Motorola 68HC11) handling the robot’s sensors.

2.2. The route-learning system

As already discussed the SOFM provides a method of topological mapbuilding and identification of perceptual landmarks suitable for localisation tasks in autonomous mobile robots (see also [6, 7, 2, 9]).

The association between locus and action which is necessary for route learning can be achieved by extending the input and weight vectors to include the required actions (see fig 3). During route training, actions form part of the input vector and thus become part of the weight vector of the trained units. During route recall, the action part of the input vector is set to zero, and the required action can be retrieved from the weight vector of the winning unit. Used in this way the SOFM is an associator between sensory information and action (see [2]).

Figure 3 shows the general principle of the SOFM as associator.

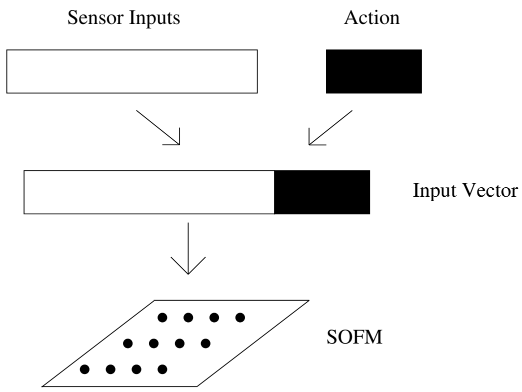

Figure 3. SOFM as associator

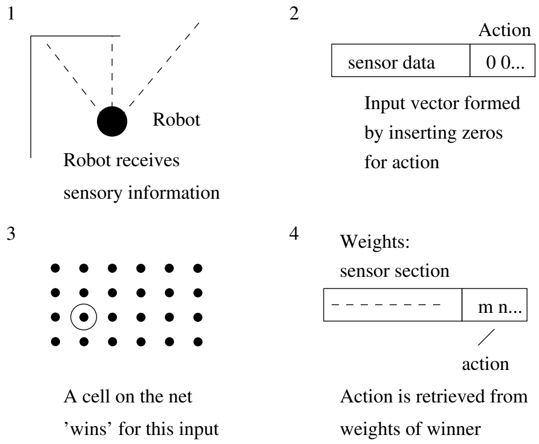

Figure 4. Action retrieval sequence

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.

The winner is the cell whose weights most closely match the sensor data; 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 final step is then to retrieve the action from the action section of the winner’s weight vector.

There are therefore two phases of operation:

1. Training phase - action and sensory information are

combined, and the winners weights and those of its neighbours updated.

2. Recall phase - sensory information alone fed to network (zeros for action), and action retrieved from weights of the winner (no weight update takes place in the recall phase).

2.2.1. The input vector

The sensor section of the input vector contains 22 components, eleven integer range values between 0 and 255 inches obtained from the sonar sensors indicated in figure 6, and eleven binary values obtained from the infrared sensors shown in figure 6 (see also figure 5).

Sonar readings (each between 0 and 255")	IR readings (0 - no object, 1 - object detected)
11 components	11 components

Figure 5. Sensor section of the input vector

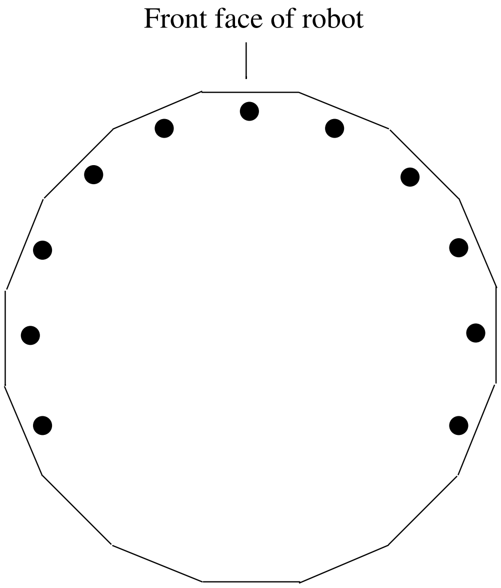

Figure 6. Sensors used

The robot's motion is given by translational and rotational velocity. For these experiments these are set to a constant value (a different constant for each of the two velocities), with the direction of rotation being indicated by the sign of the rotational velocity (positive indicating left turn, negative right). The possible motions of the robot in our navigation experiments are:

- Forward (by constant velocity)
- Right (by constant velocity)
- Left (by constant velocity)
- Combination of forward and left or right

with the required motion being read from the joystick during training. This motion is passed on to the robot's motors using the appropriate function, and is recorded in the action section of the input vector as follows:

1 0 - forward
0 1 - left
0 -1 - right
1 1 - forward and left
1 -1 - forward and right

The action section combined with the sensor section gives an input vector with a total of 24 components.

In order to achieve a balanced interpretation of each section of the weight vector (sonar, IR and action) they are normalised individually.

2.2.2. Network Parameters

The size of the network used in these experiments was 15*15 cells with a learning rate of 0.2 and a neighbourhood size of 1. The network is torus-shaped to avoid edge effects, and the weights of the cells are initially set to random numbers using a random number generator.

3. Experimental results

"FortyTwo" was trained to learn and navigate along four different routes. In each experiment the robot navigated along the route successfully for five consecutive runs. For each route the experiment was repeated four times, with a freshly initialised network on each occasion (i.e. twenty successful completions of the entire route for each of the four routes trained). For each of these trials the number of training visits required at each turning location is recorded, this is the number of times that the robot needed to be trained at that location before autonomous operation was achieved at that position within the route. This number is the total number of training visits to the particular location in the course of the experiment.

The total training time is also given for each trial, this is the time taken until the end of the robot's first totally independent circuit of the route without error (with four subsequent circuits also being completed successfully).

3.1. Results

Route 1: Figure 7 shows the first route on which FortyTwo was trained in the laboratory.

The route is simple and the average training time was approximately 18 minutes. A network response for this route is shown in figure 8 (taken from trial 1). This figure shows the change in the position of the maximally excited network unit as the robot moves through the environment. The results for this route are given in figure 9.

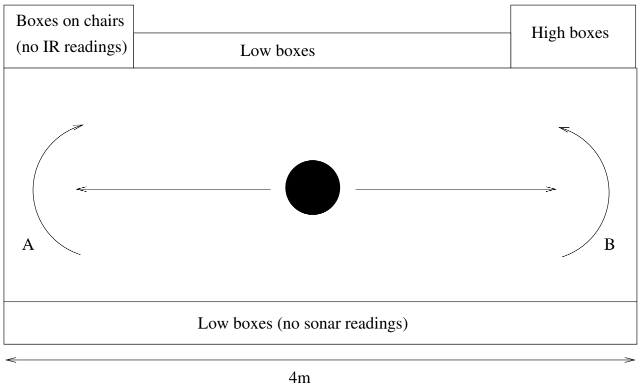

Figure 7. Route 1

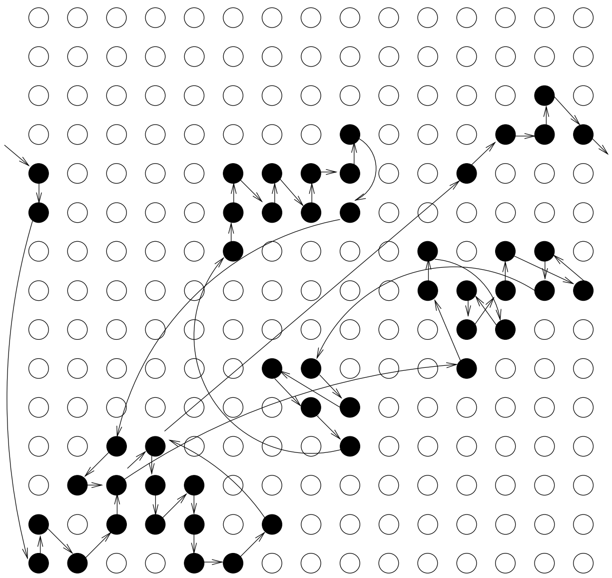

Figure 8. A network response (route 1, trial 1)

	Location	Training visits required
Trial 1:	A	4
	B	5
	Approx. total Training time	14’
Trial 2:	A	3
	B	5
	Approx. total Training time	19’
Trial 3:	A	3
	B	4
	Approx. total Training time	15’
Trial 4:	A	3
	B	5
	Approx. total Training time	21’

Figure 9. Route 1 results

Route 2: (figure 10)
Again a very simple laboratory route, average training time approximately 23 minutes. A sample network response for this route (taken from trial 1) is shown in figure 11. Here the robot started and finished in region ‘A’, the cells activated at each particular location are marked accordingly. The results for this route are given in figure 12.

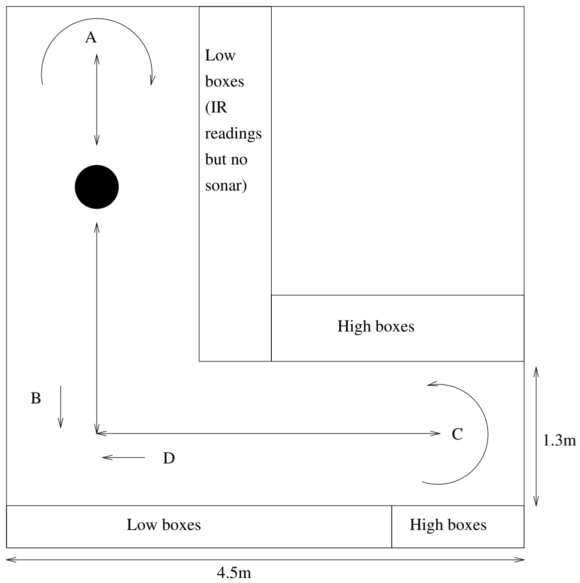

Figure 10. Route 2

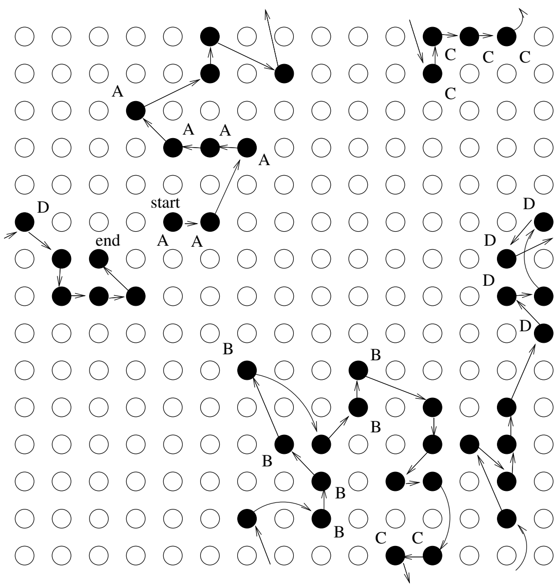

Figure 11. A network response for route 2

Trial 1:

Location	Training visits required
A	3
B	4
C	4
D	5
Approx. total Training time	25’

Trial 2:

Location	Training visits required
A	4
B	4
C	3
D	4
Approx. total Training time	22’

Trial 3:

Location	Training visits required
A	3
B	4
C	3
D	4
Approx. total Training time	24’

Trial 4:

Location	Training visits required
A	3
B	4
C	5
D	4
Approx. total Training time	20’

Figure 12. Route 2 results

Route 3: (figure 13)
This route was the first route that was attempted outside the robotics laboratory. What we have is a very simple route but one much longer than anything previously attempted in the laboratory, and with a greater number of turning locations (i.e. locations where a change of direction occurs). The average training time in these trials was approximately 36 minutes.

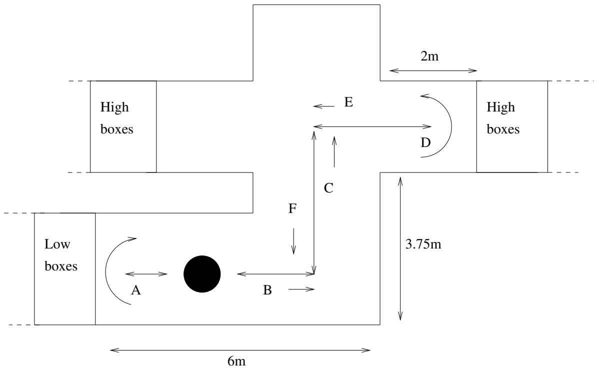

Figure 13. Route 3

Trial 1:

Location	Training visits required
A	3
B	3
C	5
D	4
E	5
F	3
Approx. total Training time	28’

Trial 2:

Location	Training visits required
A	3
B	3
C	6
D	3
E	5
F	4
Approx. total Training time	41’

Trial 3:

Location	Training visits required
A	4
B	3
C	6
D	3
E	4
F	3
Approx. total Training time	34’

Trial 4:

Location	Training visits required
A	3
B	3
C	4
D	3
E	6
F	3
Approx. total Training time	38’

Figure 14. Route 3 results

As can be seen from the results shown in figure 14, the most difficult locations for the system to differentiate were

'C' and 'E'. The cause of this was that these two locations were perceptually quite similar. Looking at figure 13 one can see that the robot is presented with similar situations at these two points on the route, in each case there is a corridor to the front and a corridor to either side of the robot. These corridors are of different length in each case but are similar enough to cause confusion in the early stages of network development.

Route 4: (figure 15)
This route followed a different format to the previous routes being more obviously circular in nature. Average training time 49 minutes.

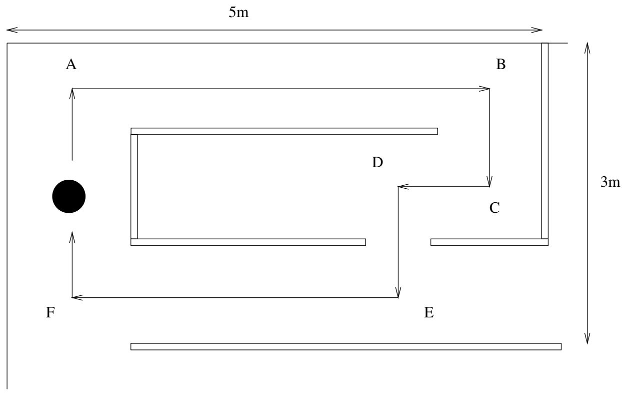

Figure 15. Route 4

As can be seen from the results in figure 16 the most difficult location to learn was D. This was because, as can be seen in figure 15, the gap through which the left turn was to be made at D was very narrow, and consequently more precision was required in the robots actions at this point to enable it to pass through the gap. This extra precision required additional training.

3.2. Fault Tolerance

A test for the fault tolerance of the system was carried out using route 4. Once the network for trial 1 had been fully trained, two of the robots sonars were disabled by replacing their readings with zeros at the corresponding positions in the input vector. The sonars that were disabled are shown in figure 17 (the decision on which sonars to disable was arbitrary).

With the sensors disabled in this way the robot was still able to complete the whole route successfully. The reason for this is that in this case the remaining sensors supply perceptual stimuli to the network that are sufficiently close to the original ones so that the learned actions can be retrieved correctly. Obviously, the more sensors that fail the higher the probability for erroneous recall due to perceptual aliasing - the navigation mechanism is dependent on distinctive sensor readings, although it can cope with a certain amount of degradation in acuity.

	Location	Training visits required
Trial 1:	A	3
	B	1
	C	2
	D	7
	E	3
	F	2
	Approx. total Training time	53'
Trial 2:	A	4
	B	2
	C	4
	D	6
	E	3
	F	2
	Approx. total Training time	40'
Trial 3:	A	3
	B	2
	C	3
	D	6
	E	4
	F	3
	Approx. total Training time	42'
Trial 4:	A	4
	B	2
	C	3
	D	6
	E	4
	F	2
	Approx. total Training time	61'

Figure 16. Route 4 results

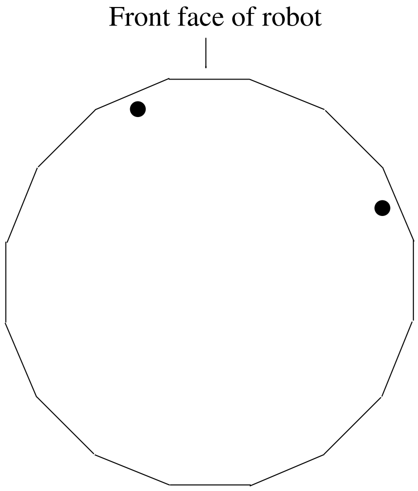

Figure 17. Disabled sonars

3.3. Generalisation

To test the generalisation ability of the system the robot was trained to perform a right turn at a given intersection, and then presented with slight variations of that intersection. Figure 18a shows the intersection at which the robot was trained, figures 18b and 18c show the two variations with which the robot was presented on completion of training. Training for the intersection took approximately four minutes, with the robot requiring to be guided around the turn three times before autonomous operation was successful (i.e. before the exit trajectory from the turn was correct).

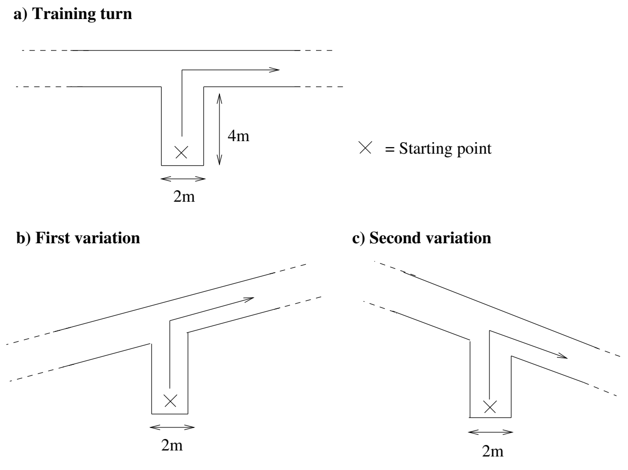

Figure 18. Generalisation

When presented with the two variations of the turn, although the robots motion was not as smooth as for the training turn the exit trajectory was correct in both cases. In actual route learning experiments we observed a similar robustness when landmarks (stacks of boxes) were shifted by up to 50cm: the robot was able to complete the routes successfully, albeit in a more wavering manner than in the unaltered case.

4. Summary and conclusion

In this paper we have presented a method for route learning in mobile robots, and tested it experimentally using a Nomad 200 mobile robot The basis of the control mechanism described is a Self Organising Feature Map (SOFM) [4]. Sensor signals from the robot's infrared and sonar sensors form the input to the SOFM. Excitation patterns developing through the process of self-organisation are associated with motor actions; the robot can be trained to follow arbitrary routes.

The route learning mechanism presented uses low level (sub-symbolic) information alone. The ability of the SOFM to map from high dimensional input space into lower dimensional space by means of a self-organising process provides a suitable way of representing the robot's environment (mapbuilding). Used as an associator the SOFM also provides an effective way of associating perception with action. Our experiments with a Nomad 200 mobile robot show that the system described has an ability to generalise on its experience, is fault tolerant and computationally cheap.

However, the generalisation property of the system can be problematic. Whilst the ability to generalise is required since the robot may find itself approaching a location from an angle different to the one on which it was trained, overgeneralisation can result in perceptual aliasing (the identical appearance to the robot's sensors of distinct physical locations). Ongoing research is addressing this problem of overgeneralisation and undergeneralisation.

References

[1] J. F. Canny. *The complexity of Robot Motion Planning*. Cambridge, MIT Press, 1988.

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

[3] P. Kampman and G. Schmidt. Indoor navigation of mobile robots by use of learned maps. In G. Schmidt, editor, *Information Processing in Autonomous Mobile Robots*. Springer Verlag, Berlin, Heidelberg, New York, 1991.

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

[5] D. Marr. *Vision - A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Co., San Francisco, 1982.

[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] N. Oreskes, K. Schrader-Frechette, and K. Belitz. Verification, validation, and confirmation of numerical models in the earth sciences. *Science*, 263:641–646, Feb. 1994.

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

[10] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Chapman and Hall, UK, 1993.