Map Interpretation in Dynamic Environments*

C. Owen and U. 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

In this paper we present a navigation system for a mobile robot that is capable of operating in dynamic environments. Mapbuilding is based on landmark detection, with landmarks being established through a process of self-organisation of the robot's sensory data. The resultant map can then be used to determine, and subsequently follow, arbitrary paths through the environment. The results of several experiments carried out with a Nomad 200 mobile robot in a dynamic environment are presented.

1 Introduction

In the field of mobile robotics there have been various approaches to mapping of the environment. Some systems use a geometric representation [1, 2], whilst others take the topological approach, whereby the environment is modeled as a graph containing nodes representing distinct locations, and pathways between locations are denoted by arcs [3, 5, 6]. However, most methods assume static environments, which can be a problem for "real world" applications where some allowance must be made for change within the robot's environment.

In addition, limitations are often imposed by the exploration strategy used in the mapbuilding process. In using wall following, for example, the type of environment that can be adequately covered is highly restricted.

In this paper we describe a system that constructs a topological map of the environment, using an approach to landmark detection based on a process of self-organisation of the robot's sensory data. This representation emerges as the robot is guided along by the user; thus generating a more generally useful map of the environment since it is the user, not the robot, who decides what areas of the environment should be covered by the map. The results of several experiments, carried out with a real robot, are also presented.

2 The Navigation Mechanism

The system consists of two main processes: *Mapbuilding* and *Map interpretation*. During the mapbuilding phase, the user guides the robot around the environment and the system constructs the topological map. In map interpretation, the robot plans and executes a path from its current location to a user specified goal.

The robot used for our experiments was a Nomad 200 mobile robot (see figure 1). In the results described here, only the sonar sensors and compass were used.

- 16 sonar sensors
- 16 IR sensors
- 20 tactile sensors
- compass
- CCD camera

Figure 1: *The Nomad 200 mobile robot.*

*Proc. 5th International Workshop on Advanced Motion Control", IEEE Press, ISBN 0-7803-4484-7, 1998.

2.1 Mapbuilding

The representation used to describe the environment takes the form of a vector map. Here, distinct locations, or landmarks, are connected by vectors denoting the distance and direction between them. Figure 2 shows the form of the vector map (for discussion on using a topological mapping of this type in preference to a metric representation see [7]). Landmarks within the

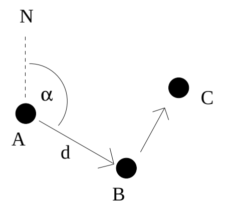


Figure 2: *The vector map. Three points 'A', 'B' and 'C' represent particular landmarks within the environment, the arcs connecting the nodes record the compass direction, α, and distance, d, between these landmarks.*

environment are selected by the system itself through a process of self-organisation of the robot's sensory data. For the experiments described in this paper, a Reduced Coulomb Energy (RCE) [9] network was used for this purpose. With this method, mapping is achieved by performing an unsupervised clustering of the robot's perceptual data as it moves through the environment. The resultant classifications can be seen as 'perceptual landmarks' within the environment.

The approach of self-organisation to landmark detection was chosen for two reasons. Firstly, user defined landmarks tend to be rather simplistic. This is due to the inherent difficulty for the human designer in interpreting the world using the robot's relatively impoverished sensors. Thus, the type of environment that can be mapped using this method is restricted. Secondly, since a generalisation over perceptions is afforded in using such clustering techniques, this mechanism gives a robust, noise tolerant, method of landmark detection.

The RCE network. The RCE-Classifier is a method of classification based on self-organisation. Each class is represented by a *representation vector* (R-vector). Training the RCE-Classifier involves determining the R-vectors.

When a pattern is presented to the classifier, the input is compared to each of the already existing R-vectors, using some form of similarity measure (e.g. dot product) in order to determine the R-vector of highest similarity with the currently perceived pattern. If the similarity between the input pattern and the 'winning' R-vector is within a pre-determined threshold then the input pattern belongs to the class of this winning R-vector. If the similarity is outside the threshold then the input pattern becomes a new R-vector. Thus the boundaries of classes are determined by the nearest neighbour law. Figure 3 shows an example for a two-dimensional input vector.

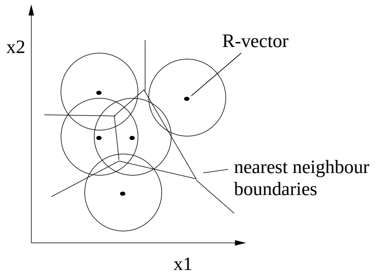


Figure 3: *RCE-Classifier, two-dimensional example. Each 'dot' in the diagram represents an R-vector. The circles surrounding each R-vector denote the 'threshold area' within which an input pattern must fall in order to belong to the corresponding R-vector's class. In the case of patterns falling within more than one threshold area, the nearest neighbour law applies.*

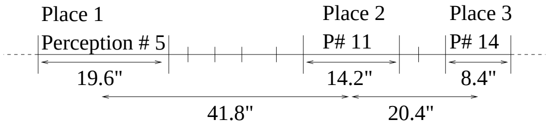
In our experiments the input vector to the RCE network consisted of the sixteen readings of the robot's sonar sensors. This input vector was normalised and the dot product of the input and R-vector was used as the measure of similarity. The threshold for adding new R-vectors was fixed at 0.9.

In these experiments the robot's onboard compass is used to align the turret (which contains the sonar sensors) constantly to compass north. This ensures that sensory perceptions are dependent on the robot's location alone, not on the robot's steering orientation.

Building the vector map. The vector map is built by the robot as it is led around the environment by the user. Movement of the robot is restricted to forward or turn, and the robot travels at a constant speed of 4 inches/sec.

As the robot moves forward it continuously takes in all 16 sonar readings and processes them using the RCE-classifier, to generate the 'perceptual landmarks'. Each time the perception changes the robot takes note of the distance travelled since the start of the last perception. To aid in re-detection, only landmarks that persist for a travelling distance of over 4 inches are added to the vector map.

Each place node of the vector map consists of a perception number (as assigned by the RCE-classifier) and a list of the places connected to that node. For each connected place, details of compass direction, distance, and size of perceptual area are stored. Each link is recorded bi-directionally. Figure 4 shows an example of three connected place nodes and the corresponding vector map.



Place 1 Perception #: 5 Number of links: 2 Link 0: Place: 0 Direction: 31 Distance: 338 Size: 224 Link 1: Place: 2 Direction: 211 Distance: 418 Size: 142	Place 2 Perception #: 11 Number of links: 2 Link 0: Place: 1 Direction: 29 Distance: 418 Size: 196 Link 1: Place: 3 Direction: 209 Distance: 204 Size: 84	Place 3 Perception #: 14 Number of links: 2 Link 0: Place: 2 Direction: 26 Distance: 204 Size: 142 Link 1: Place: 4 Direction: 207 Distance: 150 Size: 144

Figure 4: *Three connected place nodes and the corresponding vector map. Start and end points for each perception are denoted by vertical lines along the path. Here only three 'landmarks' (labelled* 5, 11 *and* 14*) are persistent enough to warrant place node generation, these landmarks have been given the corresponding place labels,* 1, 2 *and* 3 *in the vector map. Distance and size are measured in tenths of inches.*

2.2 Map interpretation

The object of the map interpretation phase is to plan a route from the current location to an arbitrary, externally specified goal location using the vector map. The basis of the planning mechanism is the 'best-first search' algorithm, which is used in this instance to determine the *shortest* known path between the current location and the goal location.

In previous experiments (see [7]), the robot was sometimes unable to find a place at the location given by the map. This phenomenon can be particularly prominent at locations near to junctions or cluttered areas where even small deviations in the robot's position and/or orientation can substantially alter the robot's perception. This effect can be alleviated somewhat by allowing the robot to gain more 'experience' of the environment, thus giving opportunity for acquiring alternative perceptions (and thus alternative paths) for locations where perception is problematic. This can be achieved by guiding the robot around the environment in such a way that multiple visits are made to each physical location, thereby generating multiple path entries in the map. In this way, if the robot is unable to locate a particluar node, the robot might plan another route along a similar trajectory via an alternative node.

However, rather than re-plan an alternative route immediately on failure to find a place node, a better approach would be to make a more concentrated effort to locate the lost node. To this end, the basic mechanism was extended to perform a simple search strategy whenever the robot was unable to locate a node. The search strategy begins by returning the robot to the previous node. Two attempts are then made to find the lost node by searching along trajectories 10° to either side of the initial trajectory.

2.3 Perceptual aliasing

One of the problems commonly encountered in systems that use perception as part of their mechanism is that of *perceptual aliasing*, i.e. distinct locations within the environment appearing identical to the robot's sensors. In these experiments, perceptual aliasing is resolved by a process of exploration. Whenever the robot encounters a place whose perception matches one or more places recorded in the map, the links for each candidate place are explored. A match with a recorded place is assumed if all the links are correct, otherwise a new place node is created. An exception to this behaviour is made when the current perception is 'expected' i.e. when moving from one place already recorded on the map to another along a pre-recorded link.

3 Dynamic environments

With the system as described so far, changes to the environment that occur subsequent to the mapbuilding phase (i.e. during map interpretation) are not incorporated into the map. To this end, the concept of 'cost' in link traversal was introduced. Whenever a new link is added to the map it is initialised with a 'confidence value', δ, of 0.5. If subsequently a link is traversed successfully the confidence value is increased to $\alpha + (1-\alpha)\delta_{old}$, where α is the learning rate (set to 0.5 in these experiments) and δ_{old} is the old confidence value. If the robot fails to traverse a link the confidence value is reduced to $(1-\alpha)\delta_{old}$. These calculations ensure that the confidence value ranges between 0 and 1. A cost can then be calculated for each link according to $d(1/\delta)$, where d is the distance variable for the link and δ is the confidence value. The shortest path is then calculated according to the total link cost rather than total distance. All updates to confidence value are made made uni-directionally[1].

This idea is similar to the one used in [4]. However, in our scheme, the calculation of cost is dependent on the distance recorded on the link concerned. In [4], this dependence can be ignored since the distance between each node is constant (each place node is described simply by its co-ordinates as obtained from the robot's onboard odometry mechanism, the perceptual properties of a place are not encoded). This distinction is important since the decision as to the value of the distance constant must be made by a human designer, and as such is fairly arbitrary (and may need to be altered for different environments). In using the robot's perception to decide on node generation, it is the environment itself that dictates the landmarks and distances involved. Furthermore, in systems that rely on accurate odometry, correctional methods are required to counter the effects of drift. Such mechanisms restrict the size of environment that can be mapped. In [4], for example, the robot must periodically return to a 'home' location for re-calibration of its odometry, correctional methods such as this are not necessary in the system presented here.

4 Experimental results

In this section we present the results of experiments carried out in a simple environment within the computer science building at Manchester University. Figure 5 shows the environment used in the experiments, along with the map generated by the system on completion of the mapbuilding stage.

Trial 1:
The setup for the first experiment is detailed in figure 6. The robot was placed at position A (node 27) within the environment with the task of finding place B (node 16). The path along the lowest cost route (initially equivalent to the shortest length route) was now blocked. The route subsequently taken by the robot is also shown.

Initially the robot attempts to follow the lowest cost route. However, due to a change in perception around the area of the new barrier, the robot is unable to locate one of the nodes along the path (node

[1] In the mapbuilding stage, an assumption is made that all links can be traversed in either direction. This is a time saving device that will, in general, be true. If subsequently links are found to be uni-directional (e.g. with doors that open one way only), then the cost of traversal in the 'wrong' direction would be adjusted accordingly.

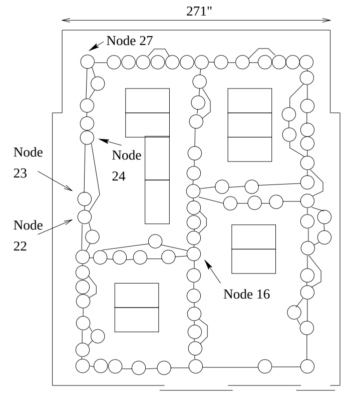


Figure 5: *The environment in which the experiments took place. The corresponding map generated by the system is also shown (each circle indicates a place node on the map). The rectangular objects are tables within the environment. The robot was driven around the environment in such a way that each place was visited at least twice. Node numbers that are pertinent to the experiments are marked accordingly.*

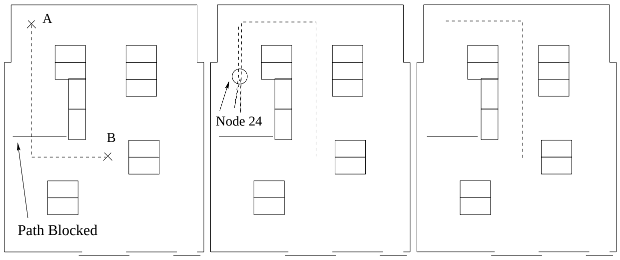


Figure 6: *The diagram on the left shows the environmental setup for the first experiment. Notice the blocked corridor at centre left of the figure. The robot was placed at location A and the task was to travel to location B. The lowest cost route before the corridor was blocked is shown as a dotted line. The middle diagram shows the route that the robot subsequently followed. The diagram on the right shows the path the robot took on being returned to position A.*

22). The robot therefore backs up to the previous location (node 24) in order to re-plan. In addition to reducing the confidence level of this link, the node that the robot was unable to locate is temporarily flagged as a 'failed' node. This means that whilst the robot remains at node 24 no further attempt will be made to plan routes via this node. There is, however, an alternative route, along the same direction, via another node connected to node 24 (i.e. node 23). Since this is now the lowest cost route from the current node, the robot then attempts to follow this alternative path.

On following this path the robot is again unable to locate the connecting node, and again reverses back to node 24. The confidence level for this link is subsequently reduced. The robot is now in the position whereby the two connecting nodes along the lowest cost path are now flagged as failed (the robot has effectively not moved from node 24 since it was unable to locate either node). Consequently, the robot generates and follows a path via the start node, as shown in figure 6.

The robot was then placed back at position A and instructed to perform the same task. This time, rather than planning a route along the blocked corridor, the robot followed a new path immediately (see figure 6). At this point this new path represents the lowest cost route between position A and position B.

Trial 2:
For this experiment, the map generated from trial 1 was utilised. The robot was again placed at position A, and instructed to find position B. In this environment two additional corridors were blocked. The path taken by the robot is shown in figure 7.

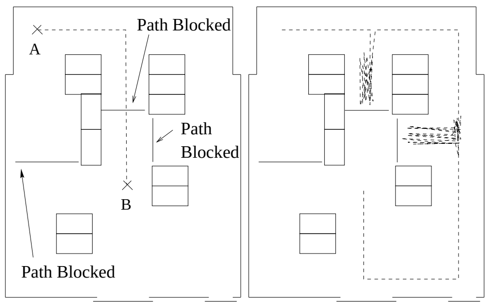


Figure 7: *The diagram on the left shows the environmental setup for the second experiment. Notice that three corridors are now blocked. The robot was placed at location A and the task was to travel to location B. The lowest cost route is shown as a dotted line. The diagram on the right shows the route that the robot subsequently followed.*

The first corridor that the robot encounters in this trial has two alternatives for traversal. However, unlike the situation in trial 1, the robot does not return to the same node on failure. Consequently oscillation occurs between the alternative paths until their cost is prohibitive, thus forcing the robot to plan an alternative route that leads out of the corridor. A similar situation arises in the second corridor.

On completion of its task the robot was placed back at position A and location B was again selected as the goal. The diagram on the left in figure 8 shows the route generated by the robot from this position.

Initially this result may seem counter-intuitive; at

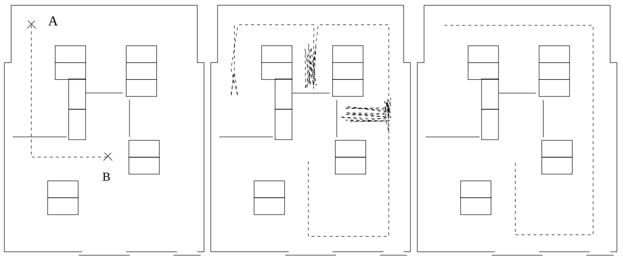


Figure 8: *The diagram on the left shows the path initially attempted by the robot on being returned to location A. The middle diagram shows the path that the robot subsequently followed. On the right is the path generated and followed by the robot on the third attempt.*

the beginning of this trial this path was rejected, and the robot was subsequently able to follow an alternative route to the goal. In general, if the robot has to re-plan several times in order to reach a goal (as is the case here), then the possibility arises that the total cost of traversal via the 'correct' route may be higher than that of paths rejected along the way. If this is the case, then it is only through experience that the robot is able to learn the correct path from start point to goal.

The route subsequently taken by the robot is also shown in figure 8. This path was generated again when a second attempt was made by the robot to reach the goal. Only on the third attempt was the cost high enough along this route to force the system to generate the 'correct' path from start to goal location, ignoring the 'dead end' corridors along the way.

Trial 3:
For this experiment the map generated during the initial mapbuilding stage was used. Figure 9 details the initial setup. The robot was placed at position A (node 16), with place B (node 24) being selected as the goal. The initial path selected by the robot is also shown.

On attempting to traverse this path, an oscillation occurs between alternative routes until cost forces the robot to generate an alternative route via the starting point. On completion of its task the robot is placed back at position A, where it subsequently generated and followed a path along the physically shortest route.

It is interesting to note that, at least from the robot's perspective, we have in effect blocked two corridors with the barrier in figure 9. On attempting to traverse to place B the robot was unable to locate nodes at the junction near the barrier. This was due to

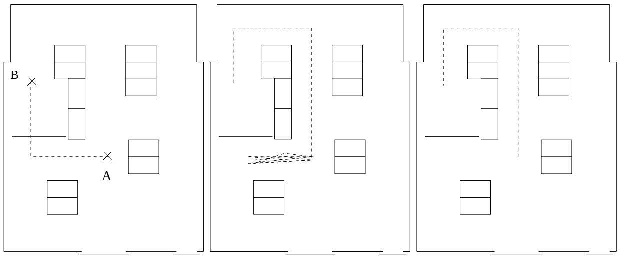


Figure 9: *The diagram on the left shows the path generated by the robot on being placed at location A with the task of locating B. The middle diagram shows the path that the robot subsequently followed. On the right is the path generated and followed by the robot on the second attempt.*

a change in perception in the area around this barrier. Since the nodes at the junction are now 'lost' to the robot, we have effectively also blocked the corridor below the barrier in figure 9. This situation is, perhaps, an inevitable consequence of using perception in the mapbuilding process, and provides an example of how different the robot's perception of the world is to our own. A potential solution to this problem is discussed in section 5.

5 Summary and Conclusions

In this paper we have presented a navigation system for a mobile robot based on landmark detection. Landmarks are selected by the system itself through a process of self-organisation of the robot's perceptual data. An initial phase of mapbuilding, guided by the user, is followed by a 'map interpretation' phase, whereby the robot plans and executes paths between locations within the environment. During the map interpretation phase the system is able to detect and incorporate changes to the environment into its map.

At present the system will fail if all known paths to a goal are blocked. In addition, unnecessary detours may be forced due to physically 'open' routes becoming perceptually blocked (see discussion, trial 3). Both these situations could be solved by allowing the user to perform additional mapping with the robot. Alternatively, the robot could use a simple exploration strategy to further map the environment whenever it wasn't involved in path finding. Thus, the need for additional human intervention may be reduced, whilst retaining the advantage gained in performing an initial user guided mapbuilding stage.

At present not all the information contained in the map is utilised. It may be possible for the system to deduce 'short cuts' within the environment by calculating a straight line path between the current location and a goal (or sub-goal) location. Such a behaviour could form the basis of a 'map-based' exploration mechanism.

The environment used in these experiments is of a simple, box world, type. Although the principle of navigation based on perceptual landmarks in 'real world' situations has been shown to be viable [8], further experimentation is required with the current system in such situations.

References

[1] P. Kampman and G. Schmidt, "Indoor Navigation of Mobile Robots by use of Learned Maps." In Schmidt, G., editor, *Information Processing in Autonomous Mobile Robots,* Springer Verlag, Berlin, Heidelberg, New York, 1991.

[2] T. Knieriemen and E. von Puttkamer, "Real-time control in an autonomous mobile robot." In Schmidt, G., editor, *Information Processing in Autonomous Mobile Robots,* Springer Verlag, Berlin, Heidelberg, New York, 1991.

[3] A. Kurz, "Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data," *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics,* Vol. 26, No. 2, pp. 233-242, 1996.

[4] B. Yamauchi and R. Beer, "Spatial Learning for Navigation in Dynamic Environments," *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics,* Vol. 26, No. 3, pp. 496-505,1996.

[5] U. R. Zimmer, "Robust World Modelling and Navigation in a Real World," *Neurocomputing,* Vol. 13, No. 2-4, 1996.

[6] M. J. Mataric, "Integration of representation into goal-driven behaviour-based robots," *IEEE Transactions on Robotics and Automation,* Vol. 8, No. 3, 1992.

[7] C. Owen and U. Nehmzow, "Landmark-Based Navigation for a Mobile Robot," To appear in: *Proc. Simulation of Adaptive Behaviour '98,* MIT Press, 1998.

[8] C. Owen and U. Nehmzow, "Route Learning in Mobile Robots through Self-Organisation," *Euromicro workshop on advanced mobile robotics,* IEEE Compute Society, ISBN 0-8186-7695-7, 1996.

[9] D. L. Reilly and L. N. Cooper and C. Erlbaum, "A neural model for category learning," *Biological Cybernetics 45,* pp. 35-41, 1982.