ELSEVIER

Robotics and Autonomous Systems 33 (2000) 223–242

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Robot navigation in the real world: Experiments with Manchester's *FortyTwo* in unmodified, large environments

Ulrich Nehmzow*, Carl Owen[1]

Department of Computer Science, University of Manchester, Manchester M13 9PL, UK

Received 11 June 1999; received in revised form 23 February 2000
Communicated by F.C.A. Groen

Abstract

Mobile robot navigation under controlled laboratory conditions is, by now, state of the art and reliably achievable. To transfer navigation mechanisms used in such small-scale environments to applications in untreated, large environments, however, is not trivial, and typically requires modifications to the original navigation mechanism: scaling up is hard.

In this paper, we discuss the difficulties of mobile robot navigation in general, the various options to achieve navigation in large environments, and experiments with Manchester's *FortyTwo*, which investigate how scaling up of navigational competencies can be achieved. We were particularly interested in autonomous mobile robot navigation in unmodified, large and varied environments, without the aid of pre-installed maps or supplied CAD models of the environment. This paper presents a general approach to achieve this.

FortyTwo regularly travels the corridors of the Department of Computer Science at Manchester University, using topological maps, landmarks, low level "enabling behaviours" and active exploitation of features of the environment. Experimental results obtained in these environments are given in this paper.

Keywords: Mobile robots; Route learning; Map building; Self-organisation; Large-scale experiments

1. Introduction

The ability to navigate is arguably the most important competence for a mobile agent. Consequently, mobile robot navigation has been the focus of considerable research effort recently.

In this paper, we present a general approach to indoor mobile robot navigation that facilitates navigation in unmodified environments of several hundred square metres size, using simple sonar and odometry sensors only. Our approach does not require the a priori installation of a map, or the use of an externally supplied CAD model of the robot's environment; it has been shown to be effective through experiments with a Nomad 200 robot in a wide range of different, large environments.

One aspect of the work presented here is that of *scaling up*. We found that a navigation mechanism that was successful in laboratory-sized environments could not be applied to larger environments straight away, but had to be augmented by additional enabling

* Corresponding author.
E-mail address: ulrich@cs.man.ac.uk (U. Nehmzow).
[1] Both authors are principal authors, with names given in alphabetical order.

0921-8890/00/$ – see front matter
PII: S0921-8890(00)00098-1

behaviours. It was the combination of the original behaviour and these enabling behaviours that made the robot succeed in navigation tasks in large environments.

1.1. Motivation

Having achieved route learning and route following in laboratory environments, we were interested in scaling up our approach — based on topological maps, autonomous map acquisition and landmark identification [14–16] — to achieve

- autonomous mobile robot navigation,
- which is independent from pre-installed maps or other navigational information,
- succeeds in unmodified environments, not containing any artificial landmarks such as beacons, induction loops or markers,
- facilitates navigation over distances of several hundred metres in populated environments,
- whilst being robust in the presence of noise and dynamic changes to the environment.

"Success" for the purpose of this paper means the ability to complete a high percentage of arbitrarily selected routes within the target environment. In all but one experiment, 100% of requested routes were successfully completed. Failures occurred only in a cluttered workshop environment containing metal objects that affected the robot's compass.

To investigate the "scaling up" of our navigation system presented in [16], we conducted experiments in the busy corridors of Manchester's Department of Computer Science, modifying our initial laboratory-scale navigation mechanism to achieve reliable navigation under the conditions listed above. The results show that robot navigation rests on a number of independent behaviours and competencies, whose interactions produce reliable and robust navigation.

1.2. Options

The fundamental competence in navigation is the ability to localise. Without knowing the current position, it is impossible to plan any paths to other locations.

The simplest option to achieve this, path integration through odometry, is not feasible for anything but laboratory experiments, due to the accumulation of drift error [1]. To exacerbate the problem, this drift error is incorrigible without resorting to mechanisms other than dead reckoning, such as, e.g. landmark identification (as in [20]).

One method that works reliably over long distances in the presence of noise and some variation is to modify the environment by installing artificial landmarks such as beacons, induction loops or visual markers. One example of many is the system by Kleeman [5].

If the modification of the target environment is either undesired (e.g. due to cost, interruption of work processes, or too short a cycle time to warrant such modification) or even impossible (e.g. due to building restrictions), the only remaining option is that of navigation by natural landmarks — a common approach taken by navigating humans or animals.

1.3. Navigation by natural landmarks

When we speak of "landmarks" or "perceptual landmarks" in this paper, we mean features of the environment that are detected by the robot's sensors (sensory patterns or "sensory signatures"), rather than sparsely distributed, conspicuous locations that are commonly identified as landmarks by humans.

The first question to consider is whether landmarks should be identified by the human operator, a priori, or by the robot, as it moves through its environment. We contend that it should be the latter on the grounds that human and robot perception of the same environment differ so radically that there is a risk that either landmarks are selected that are hard to detect by robot sensors or easily detectable landmarks are overlooked by the operator because they appear inconspicuous to a human observer.

To give an example of this fact: Identifying landmarks in an office corridor, most human observers rate *doors* as highly suitable landmarks. In the case of sonar sensors, however, our experiments show that it is not doors that are the easiest to detect, but the slightly protruding *door frames* — rather inconspicuous to humans, but highly visible to sonar sensors.

A second major consideration when navigating by natural landmarks is to find a generalised representation of landmarks that allows identification of the landmarks regardless of viewing angle, lighting conditions, etc. and yet preserves the distinguishing

features between different landmarks, to keep the degree of confusion between landmarks (perceptual aliasing) low. As before, we argue that the best approach to take here is to use mechanisms of self-organisation rather than pre-defined classifiers. By *acquiring* the model, the robot is better able to exploit those features that are visible to its sensors and to change its internal representation according to changes in the environment around it.

1.4. The approach

Following these considerations, we have built and tested a mobile robot navigation system that uses topological maps of categorised sensory signatures of natural landmarks to achieve localisation, path planning and path following. Landmarks are categorised using self-organising artificial neural networks.

The mechanism has been implemented on a Nomad 200 mobile robot and was tested in unmodified, changeable and populated environments of several hundred square metres size. The results of experiments conducted in five different environments, under a broad spectrum of environmental conditions, confirm that the method is not restricted to one particular environment and to navigation over short routes only, but is suitable for autonomous mobile robot navigation in large environments of widely different perceptual properties.

2. The fundamental map building and map interpretation mechanisms used

We will begin by describing the basic system used in the initial experiments described in [17]. In Section 3, we will then describe the extensions required in order for the system to operate in office type environments covering several hundred square metres.

The system consists of two main operational phases: *map building* and *map interpretation*. During the map building 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 using the previously acquired map.

The robot used for our experiments was a Nomad 200 mobile robot (see Fig. 1). This robot is equipped

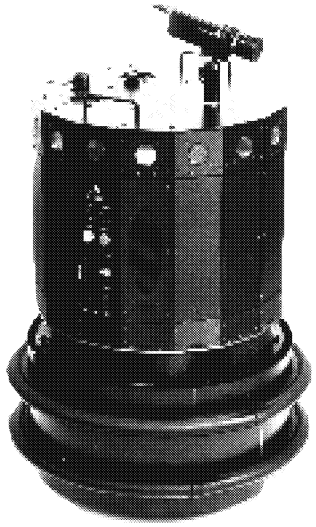

Fig. 1. The Nomad 200 mobile robot.

with 16 ultrasonic range finding sensors (range up to 6.5 m), 16 infrared (IR) sensors (range up to 60 cm), 20 tactile sensors, a flux gate compass and a monochrome CCD camera. In the experiments described here, only the sonar sensors and the compass were used.

2.1. Map building

Fig. 2 shows the form of the vector map used to represent the robot's environment. In this map, distinct locations, or landmarks, are connected by vectors denoting the distance and direction between them.

In employing a representation of this type, we restrict the use of odometry to the measurement required between locations ("local" odometry), thus limiting the global accumulation of odometry error.

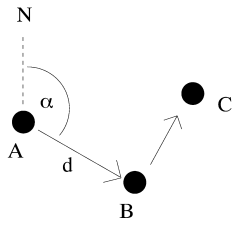

Fig. 2. The vector map. Three points A, B and C represent particular perceptual landmarks within the environment, the arcs connecting the nodes record the compass direction, α, and distance, d, between these landmarks.

Landmarks within the environment are determined by the system itself through a process of self-organisation of the robot's sensory data. In the experiments described in this paper, a reduced Coulomb energy (RCE) [19] network was used for this purpose. With this method, an unsupervised clustering of the robot's perceptual data is performed as it moves through the environment. The resultant classifications can be seen as "perceptual landmarks" within the environment.

By this technique of autonomous, user-independent landmark selection, only landmarks that are actually perceivable by the robot are used. This ensures (a) that the robot is not reliant on features it cannot actually detect and (b) all available information is used for the navigation process. In addition, since the clustering techniques used in self-organisation enable a generalisation over perceptions, this approach gives a robust, noise tolerant method of landmark detection.

2.1.1. The RCE network

The role of the classifier is to classify sensory perceptions such that noise and small perturbations do not affect the robot's ability to recognise an already mapped location. Self-organising feature maps (e.g. [6]) are a possibility and have successfully been used for this purpose [12].

However, one particular aspect of the work presented here was to scale up a navigation mechanism from laboratory-sized environments to potentially unbounded environments. A network of finite size is problematic in such environments. We have, therefore, used a growing neural network as classifier, the RCE net [19].

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. Fig. 3 shows an example for a two-dimensional input vector.

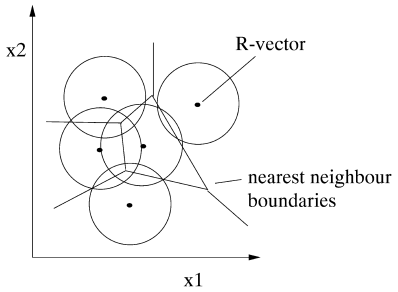

Fig. 3. RCE-classifier, two-dimensional example. Each "dot" in the diagram represents an R-vector. The circles surrounding each R-vector denote the "threshold area" within which an input pattern must fall in order to belong to the corresponding R-vector's class. In the case of patterns falling within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

In our system, the input vector to the RCE network consists of the 16 readings of the robot's sonar sensors. A smoothing procedure is applied to the input vector by delaying any large change in the input data by a pre-defined number of time steps, with the reading in question simply taking on its old value for the duration of this period. If the new data value still holds on completion of the allotted interval, it is then allowed to pass through to the classifier as part of the input vector. The purpose of this mechanism was to remove spikes from the input data. Both the amount of change defined as large and the number of time steps to skip were determined experimentally by observing data time series from several locations within the Computer Science Department building (see [17]).

The resultant input vector was normalised and the dot product of the input vector and R-vector was used as the measure of similarity. If this dot product fell below a preset threshold of 0.9, a new R-vector was added to the map.

In our experiments, the robot's onboard compass was used to align the turret (and thus the sonar sensors) constantly to local compass north. This ensures that sensory perceptions are dependent on the robot's location alone, not on the robot's steering orientation. Local deviations from true north (for instance due to metal structures in the building) obviously do not

affect the orientation mechanism as long as these deviations are consistent over time at a particular location.

2.1.2. 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 $10\,\mathrm{cm\,s^{-1}}$.

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. Since small landmarks could easily be missed on re-visiting a location, only landmarks that persist for a travelling distance exceeding 10 cm 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, because due to local anomalies in the building's magnetic field, opposite compass directions are not simply computed by adding or subtracting 180°. Fig. 4 shows an example of three connected place nodes and the corresponding vector map.

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.

Best first search is a graph search technique whereby, at each step, the node along the shortest route length estimate is expanded. Fig. 5 shows an example of this method.

Once the path has been generated, the robot takes each node in turn and attempts to find the next node along the path. Once the "correct" perceptual landmark has been found, the robot moves to the centre of the landmark by moving forward half the size of the landmark, as indicated by the vector map.

The robot aborts an attempt to move towards a landmark if the sought sensory perception is not encountered within the travel distance between the nodes, indicated by the vector map.

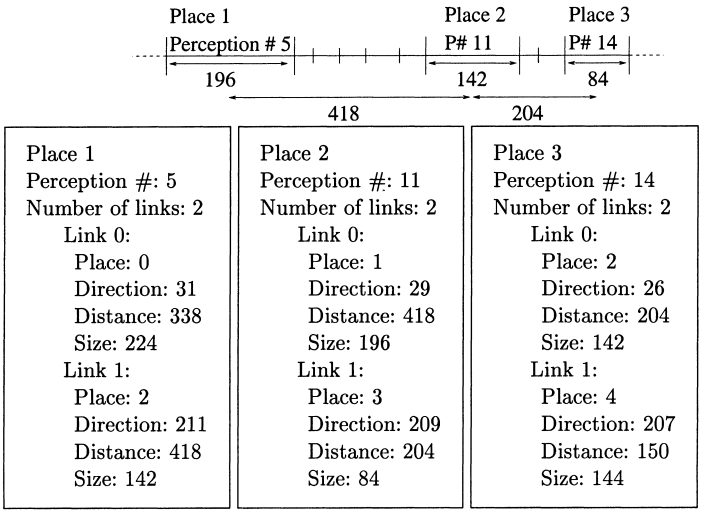

Fig. 4. Three connected place nodes and the corresponding vector map. Start and end points for each perception are denoted by vertical lines along the path. Here only three "perceptual landmarks" (labelled 5, 11 and 14) are persistent enough to trigger place node generation, and these landmarks have been given the corresponding place labels, 1, 2, and 3 in the vector map. Distance and size are measured in units of 2.5 mm.

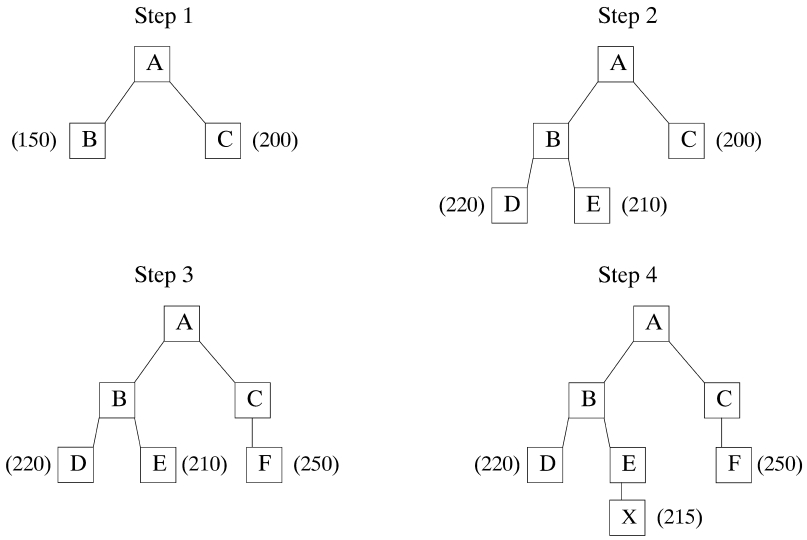

Fig. 5. An example of best first search. The search starts at node *A* and the goal is *X*. Step 1: *A* is expanded producing two nodes *B* and *C* (the path length to these nodes is shown in brackets). Step 2: The path to node *B* is the shortest and is expanded, producing *D* and *E*. Step 3: At this point the path to *C* is shortest, expansion producing node *F*. Step 4: The path to *E* is expanded, producing the goal node *X*. Since the path length to *X* is shorter than any of the competing paths from *A*, this is the shortest path from *A* to *X*.

Experimental results obtained in laboratory environments showed that the proposed method worked reliably. We were now interested to apply the method to navigation in much larger environments.

3. Scaling up: Enabling behaviours and enabling mechanisms

Several experiments conducted with our system as described so far are detailed in [17,18]. The results of these experiments showed the system to be robust and adaptable to change. However, these experiments were conducted in simple, small-scale environments. The next step was to use the mechanism in more complex environments covering much larger distances. We found that, to achieve reliable navigation under these circumstances, the robot's repertoire of enabling behaviours had to be extended in the way described below.

The first large-scale experiment was conducted on the second floor of the Computer Science Department building. Fig. 6 shows an example of such a corridor.

These corridors are fairly featureless and as a consequence perceptual landmarks tend to be large and the distances between them can be fairly high. The first problem we encountered, therefore, was that since the compass resolution is low (to within 2°), and the robot centres itself at a landmark by simply travelling the prescribed distance on encountering the required perception, the system can fail due to incorrect positioning of the robot. Fig. 7 gives an example of this problem.

3.1. The centring behaviour

Our first approach to address this problem of low-precision compass readings was to develop a better method of centring the robot once it had encountered the desired perception. Fig. 8 depicts the concept of centring.

The method we chose was to store perceptual snapshots (consisting of the 16 sonar readings of the robot) for the centre of each place, and to move the robot so as to reduce the difference between the current perception and the central snapshot. Fig. 9 gives a two-dimensional example of how a "difference vector" can be calculated between the current perception and a stored snapshot. This vector can then be used to calculate the movement required in order to reduce the difference.

Fig. 6. A typical corridor within the Computer Science Department.

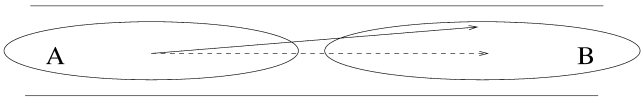

Fig. 7. Two landmarks *A* and *B* occurring within a long featureless corridor. The dotted line depicts the direction the robot should take from *A* to *B*, the solid line depicts the actual line taken by the robot due to low compass resolution. The robot is now incorrectly positioned for the start of the next stage of its path following task.

The difference vector determines which perception component values need to be increased (positive numbers) or decreased (negative numbers). In relation to the robot's actual movement, this tells us which sonar values need to be increased/decreased. An increase in sonar value translates into a move away from an object, the converse being true of a decrease in value. The robot is taken to be centralised at a landmark when the dot product between the stored and current vector reaches a threshold of 0.98 (with both vectors being normalised to unit length).

Initially, we moved the robot according to the highest absolute value in the difference vector. However, due to noise, this method gave very unsatisfactory results. A high incident angle between a sonar and an object, e.g. can cause a value much larger than the actual distance between them to be returned due to specular reflections. Errors in the input vector such as this can be corrected by the physical movement of the robot (the readings will change as the robot changes its position), but unfortunately this effect alone is not sufficient to make the mechanism reliable.

A much better result was achieved by calculating the direction of travel according to the quantity of negative or positive values in the input vector. If a higher number of negative values are present, then the

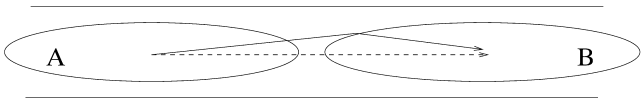

Fig. 8. The centring behaviour. Once the perception for the next landmark is encountered, the robot uses the centring behaviour to reach the centre of landmark *B* by minimising perceptual differences between stored and actual perceptions. This corrects the error introduced by low-precision compass readings.

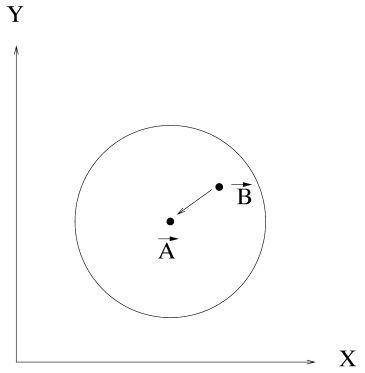

Fig. 9. Minimising perceptual difference in a two-dimensional input space. X and Y denote sensor inputs. $\vec{A}$ represents the stored snapshot for the centre of A, the circle denotes the "influence area" of the perception for which A represents the centre. $\vec{B}$ represents the current perception of the robot. The difference vector is given by calculating $\vec{A}-\vec{B}$. This vector can be used to calculate the actual direction of travel for the robot in order to minimise the perceptual difference between current perception and stored snapshot: Where a positive value is recorded in the difference vector, this translates to a movement away from the object at the corresponding sonar position. The converse is true for negative values.

robot must move forward in the direction given by the sonar at the centre of the corresponding negative values in the input vector. The same calculation is made for the converse case, but with the movement in reverse. This approach proved to be very reliable when the robot was within the perceptual area corresponding to the stored snapshot.

3.2. Wall following behaviour

Besides having to cope with alignment errors due to coarse compass resolution, we encountered a second problem that impeded navigation over large-scale distances. Where the distance between landmarks was very large, compass resolution could still cause system failure due to the robot touching a wall before the next perception was encountered, i.e. before the centring behaviour could be brought into operation.

Our approach to resolving this type of failure was to integrate an additional behaviour into the system. In the new scheme wall following can be used, where appropriate, to guide the robot between the landmarks. In this way, rather than moving blindly forward in a straight line, the robot can use the environment to direct its traversal.

In the map building phase the user now has a range of behaviours that the robot can be instructed to perform. These behaviours are: *forward*, *turn*, *wall follow left* and *wall follow right*. The *forward* or *wall follow* behaviour currently in use is recorded as an additional property of each connected node in a place node's list. Since links are created bi-directionally, the wall following behaviour is simply reversed for traversal in the opposite direction (i.e. wall following to the left between places A and B becomes wall following to the right between B and A). The wall following behaviour used in our experiments was a learned behaviour based on work carried out by Nehmzow [11]. This model was augmented by Duckett and Nehmzow [3], in work on localisation to enable the robot's turret to operate independently of the base.

In using wall following, the map building process itself can be made more autonomous. In areas where wall following is appropriate, the robot can be left to map the area by itself.

During map interpretation, rather than searching for each successive perception, the robot can now use the distance value contained in each link to travel directly to the centre of each landmark. Any slack generated by odometry error will be taken up by the centring behaviour.

3.3. Returning behaviour

In order to detect a failure to localise the sought landmark, a time limit of 1 minute for centring is set. When a detection failure occurs due to the path being blocked, e.g. the robot returns to the previous known location before re-trying. However, this is complicated by the fact that the robot will have attempted to centralise at the "failed" location. The robot's position and orientation may not allow a simple route reversal to take place.

The method we chose to return the robot to its previous location was to use the onboard odometry mechanism to calculate a direction and distance vector between current position and the previous landmark. The robot would then move according to this vector and, on completion of the required distance, use the centring behaviour to position itself correctly at the previously visited landmark.

3.4. Dealing with perceptual aliasing

One of the major problems in perception-based mobile robot navigation is that of *perceptual aliasing*, i.e. distinct locations being perceived identically.

One approach to dealing with perceptual aliasing is to increase the robot's perceptual resolution by adding and combining additional information from differing sensor modalities (sensor fusion) — see, e.g. [2,7]. Unfortunately, this type of approach cannot be guaranteed to disambiguate all perceptions, and is more useful as a tool for reducing, rather than eliminating, perceptual ambiguity.

In [13], a "history" of past perceptions is used to disambiguate locations. However, as reported by Nehmzow and Smithers [12], a variable length history would be required to solve the problem of perceptual aliasing. In addition, for correct identification a different history would be required for each different approach to a location. In [12,13], this problem is circumvented by using a fixed wall following behaviour, thus ensuring a canonical path, which still does not solve the problem that this method is also highly sensitive to misclassification error. An error in one element of the history will continue to affect location recognition until that member leaves the current perceptual history.

Other systems use positional information, derived from odometry, to disambiguate locations (e.g. [8,9,20,21]).

Our approach in the experiments detailed here was to use a method similar to that described in [9]. Here, a previously stored location is recognised as such if two conditions are met:

1. The current perceptual classification and the classification of the candidate location are the same.
2. The current positional co-ordinate is within a pre-defined tolerance of that stored for the candidate location.

Each place node in the map now has an added positional component. This component is derived from the robot's onboard odometry mechanism (but note that this positional information is used solely for disambiguation). The tolerance used in our experiments was 60 cm. On recognition of a previously mapped location, the stored co-ordinate is re-written as an average of the stored and current position (as in [8]). In addition, the robot's current position is reset to this average co-ordinate. In the experiments detailed here, this approach proved reliable.

3.5. Use of multiple snapshot coding in open environments

For the corridor environments in which we conducted the experiments reported in this paper, storing one snapshot for each landmark is sufficient. Due to the particular distance the robot keeps between itself and the wall during wall following, a central position is taken on traversal through the corridors (i.e. the robot travels equidistant from each wall). Since each landmark within a corridor can only be approached from two directions, this property of the wall following behaviour ensures that for each approach direction the central point of the landmark (at which position the snapshot is taken) will be the same.

For junctions within a corridor environment, the number of potential approach directions is greater than that for the corridors themselves. However, due to the perceptual properties of such junctions, the central point of the landmark at the junction will be roughly the same regardless of the approach direction. This means that the distance and direction measures, as stored on the map, will always be roughly correct.

In more open environments the central snapshot point may be much more variable, and storing only one snapshot may result in incorrect behaviour. Fig. 10 illustrates this point.

Our approach to the problem of variable central location is to use multiple snapshot entries in the map. Thus, in Fig. 10, we also store snapshot Q. The map entries for places 5, 6 and 20 would now be as in Fig. 11.

For the robot to position itself correctly, planning must now be carried out backwards from the goal towards the starting location. Thus, in planning a route from place 5 to 20, the snapshot generated for place 6 would be Q instead of Y. The robot would therefore centre itself at the correct position for traversal to 20 from 6.

3.6. Dealing with dynamic environments: mapping of changes

With the system as described so far, changes that occur subsequent to the map building stage are not

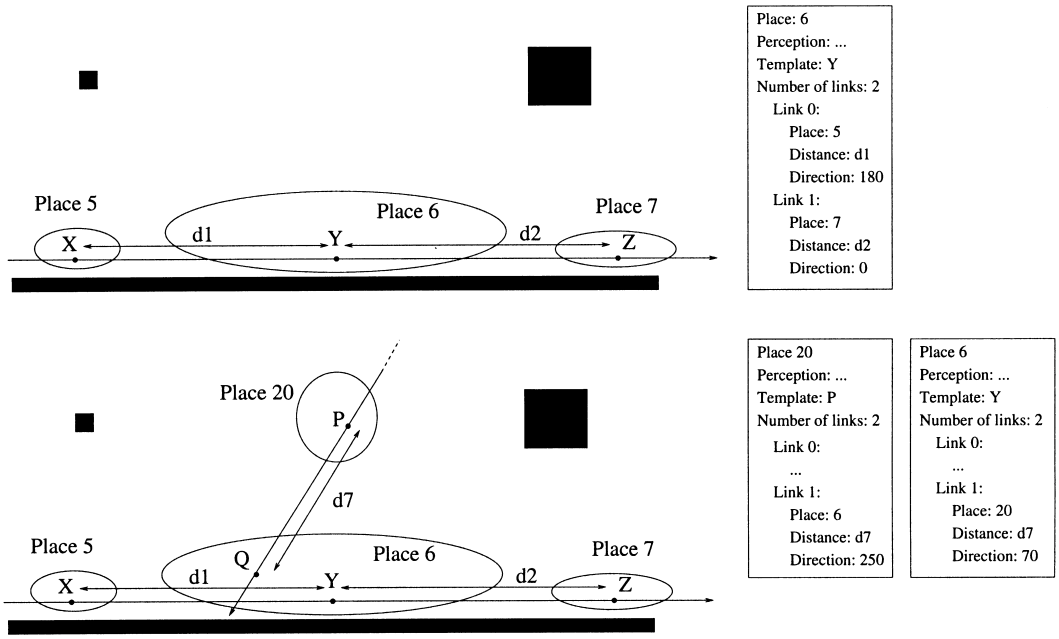

Fig. 10. Multiple approach in an open environment. In the top figure we see three place nodes, 5, 6 and 7 along with the corresponding map entry for place node 6. The snapshot stored for 6 is Y. In the figure below, the robot has just travelled from place 20 on through place 6 and a match has been made (i.e. the perception matched that of place 6 and was within the tolerance allowed for position). The corresponding map entry for place 20 is also shown along with the updated entry for place 6. If the robot used this map to plan a route from 5 to 20, the distance and direction components given for place 20 would be incorrect (the robot would centre to Y), and the robot may not be near enough to 20 for the centring mechanism to operate successfully.

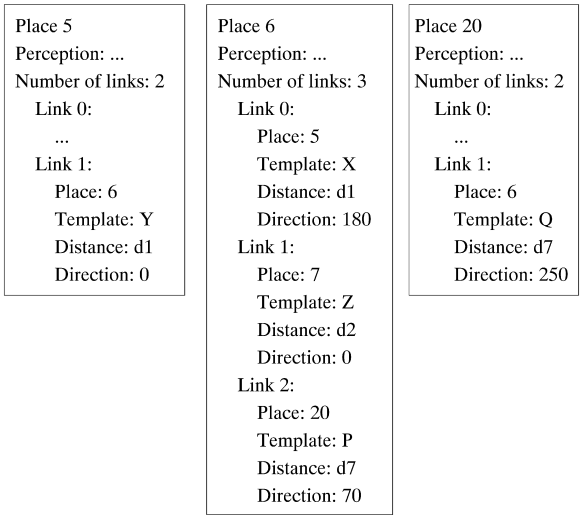

Fig. 11. New place node format. Snapshots are stored as link information.

incorporated into the map. Therefore, although the robot is able to plan alternative routes if a path becomes blocked, the blocked path will remain part of the map as there is no mechanism by which the robot can delete an obsolete link.

One approach to removing obsolete links is to introduce the idea of cost in link traversal. 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 $\alpha + (1-\alpha)\delta_{\text{old}}$, 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 $(1-\alpha)\delta_{\text{old}}$. 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 the confidence value. The shortest path is then calculated according to the total link cost rather than total distance.

This is the mechanism used by our navigation system. The idea closely follows that of Yamauchi and Beer [20]. However, the implementation details differ since the calculation of cost in our model is dependent on link length (for a further discussion see [18]).

4. Experimental results

This section details the experiments carried out with the robot at various locations within the Computer Science Department at Manchester.

4.1. Laboratory environment

The first experiment carried out with the complete system is detailed in Fig. 12.

4.1.1. Map building

In this experiment, the environment was mapped completely autonomously using the robot's left-hand wall following behaviour. On completion of one circuit of this route the robot was asked to perform path following using its map.

4.1.2. Route following

In order to test access to all of the place nodes on the map, the following sequence was presented to the system:

1. Plan and follow a path from place 0 to place 7;
2. Plan and follow a path from 7 to 10;
3. Plan and follow a path from 10 to 0.

This sequence ensured a complete circuit of the environment. This task was repeated 10 times followed by another 10 circuits in the opposite direction. Each circuit was completed successfully with all snapshot matching occurring at first attempt.

To assess the system's ability to operate in dynamic environments, a barrier was placed between nodes 0 and 12 (see dotted line in Fig. 12). The robot was then given the task of traversing from nodes 0 to 9.

For the first three attempts, the robot tried to traverse the link between 0 and 12 (i.e. anti-clockwise from 0 to 9) before re-planning and following the clockwise route. At the fourth attempt, the confidence value for link 0 to 12 had been reduced to such an extent that the clockwise route became lower in cost, and the robot chose this route without attempting to travel along the anti-clockwise path. That route was then successfully completed.

4.2. Cluttered workshop environment

For this experiment, the mechanical workshop of the department was used. This provided a very cluttered environment to operate with various benches and

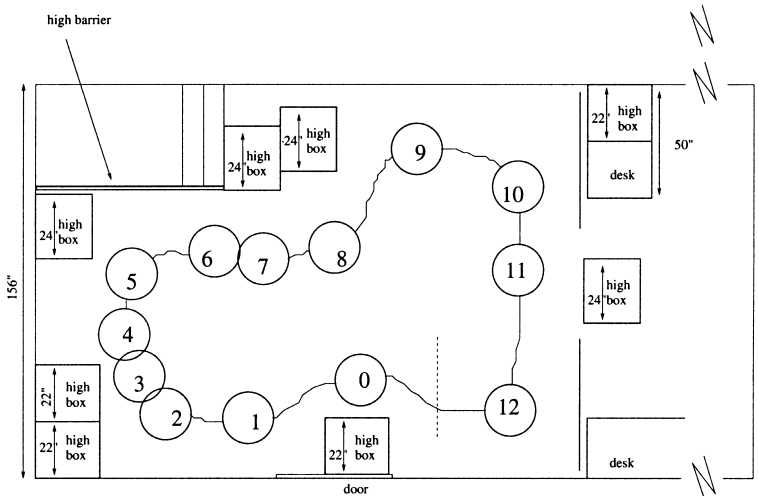

Fig. 12. A simple laboratory environment. In this experiment the robot was simply allowed to use its left-hand wall following behaviour to map the environment (i.e. no user intervention). Each circled number in the diagram represents a place node as generated by the system.

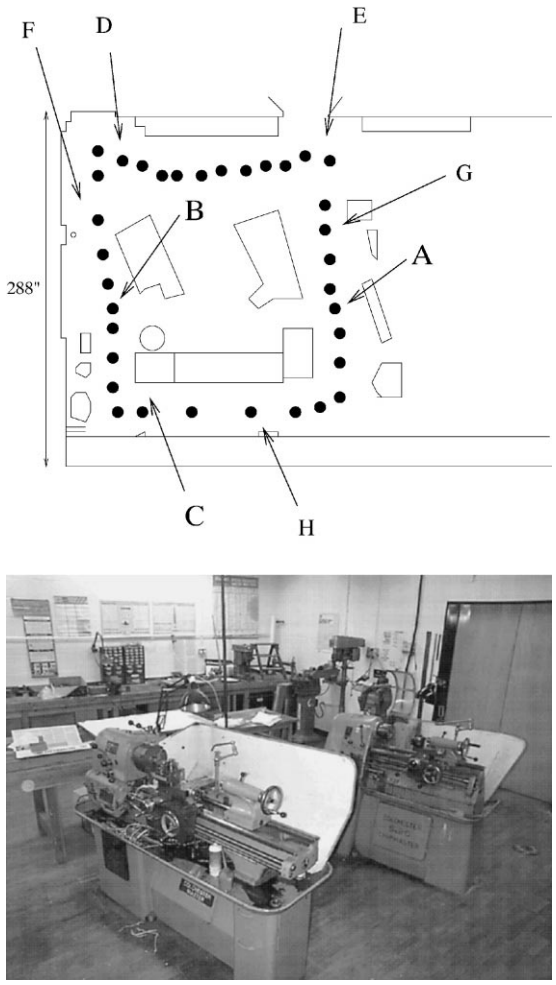

Fig. 13. A cluttered environment. In the top figure, each filled circle denotes a place node as generated by the system. The photograph below shows the actual environment (taken from location *E*).

heavy machinery scattered around. Fig. 13 shows the environment.

4.2.1. Map building

Here, the robot started at location *G* and travelled clockwise, eventually looping back to *G*. However, simple wall following was not sufficient for this task. At position *D*, the robot could not pass through the gap using wall following. Therefore, at this position, user guidance was required (i.e. the robot was turned and guided by hand through the gap).

Similarly, at location *E*, wall following would have meant that the robot would continue down along the right-hand wall in the diagram. Here, the robot was again turned and guided in order to complete the loop.

4.2.2. Route following

To investigate the robot's ability to plan and follow routes, the following routes were traversed:

1. plan and follow a path from place *G* to place *H*;
2. plan and follow a path from *H* to *F*;
3. plan and follow a path from *F* to *G*.

This sequence (i.e. clockwise from *G* to *G*) was attempted 10 times. The results of this experiment are detailed below.

Traversal from G to H. In five out of the 10 attempts to traverse from *G* to *H* the robot failed to centre at location *A*. On failure, the robot was placed manually at the landmark following that at *A*, and in each case the robot was able to successfully complete its traversal to *H*.

Traversal from H to F. In two out of the 10 attempts to traverse between *H* and *F* the robot failed to centre at location *B*. Again, on being placed at the landmark following *B* on each failure, the robot was able to successfully complete its traversal.

Traversal from F to G. In all 10 attempts, the robot was able to traverse successfully between *F* and *G*.

The robot was then given the sequence in reverse order:

Traversal from G to F. All 10 attempts from *G* to *F* proved successful.

Traversal from F to H. In four out of the 10 attempts, the robot failed to centre at location *B*.

Traversal from H to G. In eight of the 10 attempts to traverse from *H* to *G*, the robot failed to centre at *A*. On being placed at the landmark following *A* on each failure, the robot was able to successfully complete its traversal.

The failures at *A* and *B* were due to fluctuations in the magnetic field at these locations. These fluctuations affected the turret orientation, thus interfering with location recognition.

4.3. Long office corridor

Fig. 14 shows the environment used in this experiment.

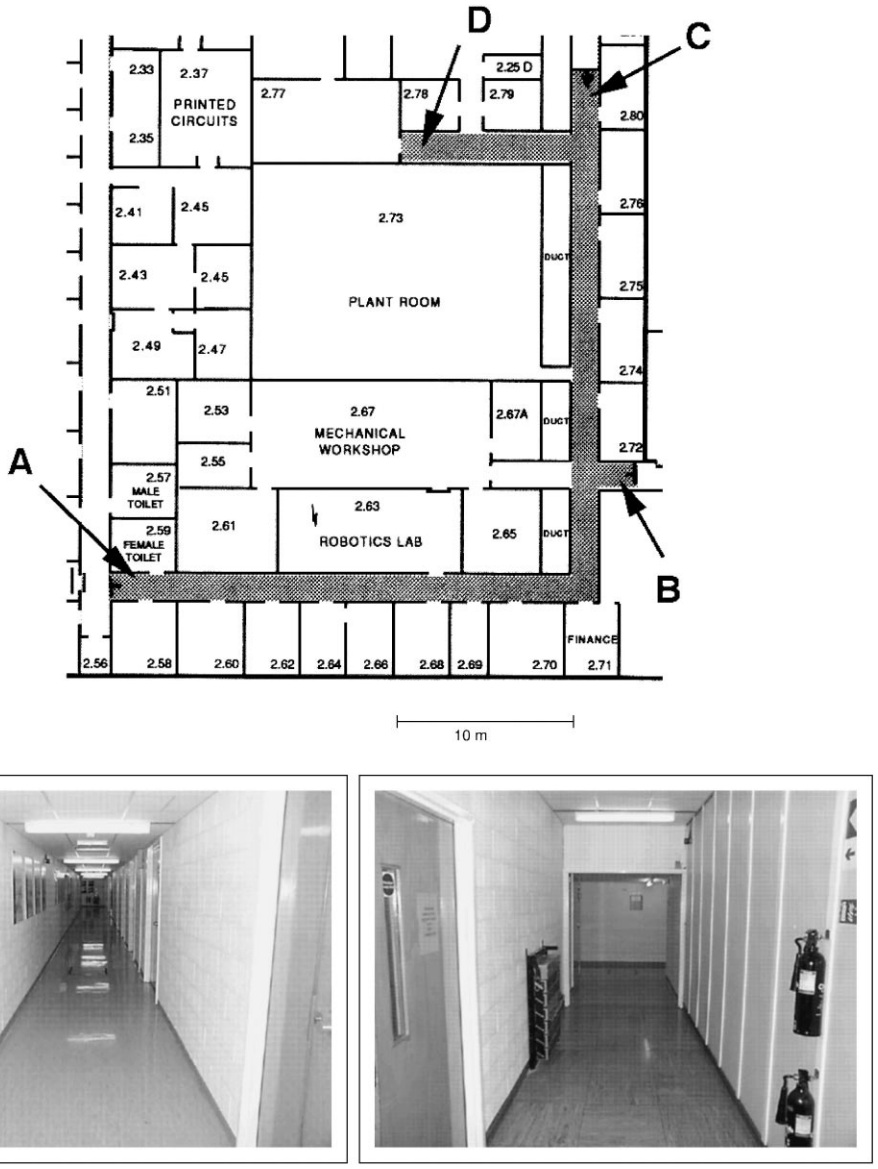

Fig. 14. A large-scale environment. The shaded area shows the enclosure mapped by the robot. The photograph on the left was taken from location *A*, on the right we see a view from *D*.

4.3.1. Map building

Starting at location *A*, the robot was allowed to perform left-hand wall following until it reached the short corridor at *B*. At this point, user guidance was used to map the short corridor at this location. The robot was then allowed to use left-hand wall following until it reached the short corridor at *C*. At this point, user guidance was used to map this corridor. Finally, the robot used left-hand wall following to complete the corridor at *D*.

4.3.2. Route following

For the first experiment, the robot was asked to plan and follow a path between *A* and *D*. Once completed,

the robot was then asked to plan and execute the same path in reverse. This cycle of forward/reverse was repeated five times. On each occasion, the robot was successfully able to traverse between the landmarks at these two locations.

A second test was then performed in order to test the accessibility of all landmarks within the environment. For this experiment, the robot was again to travel between *A* and *D*, but this time was also to visit the landmarks along the short corridors at *B* and *C*. This involved three planning and execution phases:

1. plan and execute a path from *A* to *B*;
2. plan and execute a path from *B* to *C*;
3. plan and execute a path from *C* to *D*.

As with the first experiment, the robot was then ordered to repeat the sequence but in reverse. This cycle was also repeated five times. On each occasion, the robot was able to match all snapshots on the first attempt.

In order to test the system's reliance on odometry, tests were conducted which introduced an additional odometry error into the system. In the first experiment, an error of 5% was added to the odometry readings obtained from the internal mechanism. The robot was then placed at location *D* with the task of traversing to location *A*. This path is approximately 66 m long, which equates to a cumulative error of about 3 m over the total distance.

The result of this experiment was that the robot successfully traversed the route, with its final position being correctly centred at the landmark at place *A*. The centring mechanism had effectively taken up the slack generated by the odometry error. As a consequence of the artificially introduced odometry error, the average time taken to centre the robot at each landmark rose from 16 (without odometry error) to 28 s (with error).

An additional experiment was carried out with an added odometry error of 10%. This yielded a number of failed attempts to centralise. In each case of failure, the distance between the landmarks was large and the robot had not reached the perceptual area associated with the goal landmark.

4.4. Mixture of corridor and open spaces

The environment used for this experiment is shown in Fig. 15.

4.4.1. Map building

For map building, the robot started off at location *A*. Right-hand wall following was then used until the robot reached location *B*. At this point, a turn command was issued and the robot was turned anti-clockwise to resume its right-hand wall following along the upper wall at *B*. At location *C*, the robot was guided by hand round to location *D* (there being no walls for the robot to follow). From this point, right-handed wall following was used until location *E* was reached (at which point the robot correctly identified the appropriate landmark).

4.4.2. Route following

To test path planning, the robot was placed at position *D* with the task of moving to position *A*. This was repeated five times. The path chosen in each case was along the shortest route (i.e. directly from *D* to *E* and then on to *A*). Next, five attempts were made in the opposite direction, again the shortest path was chosen. In each case, the robot traversed successfully between the locations.

A barrier was then placed at position *F*. The robot was again given the task of traversing from *D* to *A*. On the first attempt, the robot was unable to locate a landmark near to the barrier, and thus re-planned a route in a clockwise direction around the loop. This traversal was successfully completed. On being placed at *D* again, the robot exhibited the same behaviour (i.e. an attempt was made to traverse the shortest route before re-planning occurred). Again, the robot traversed the alternative route successfully. On the third occasion, the robot made no attempt to traverse by the shortest route, instead electing to travel via the longer loop — the link at *F* had correctly been removed from the map. This longer path was traversed successfully.

4.5. Open environment

For our final experiments, the environment shown in Fig. 16 was used. This environment was of a more open nature than the corridor type environments of previous experiments.

4.5.1. Map building

Here, the robot mapped the periphery partly by using wall following (right-hand) and partly by user guidance (where no walls were available for the

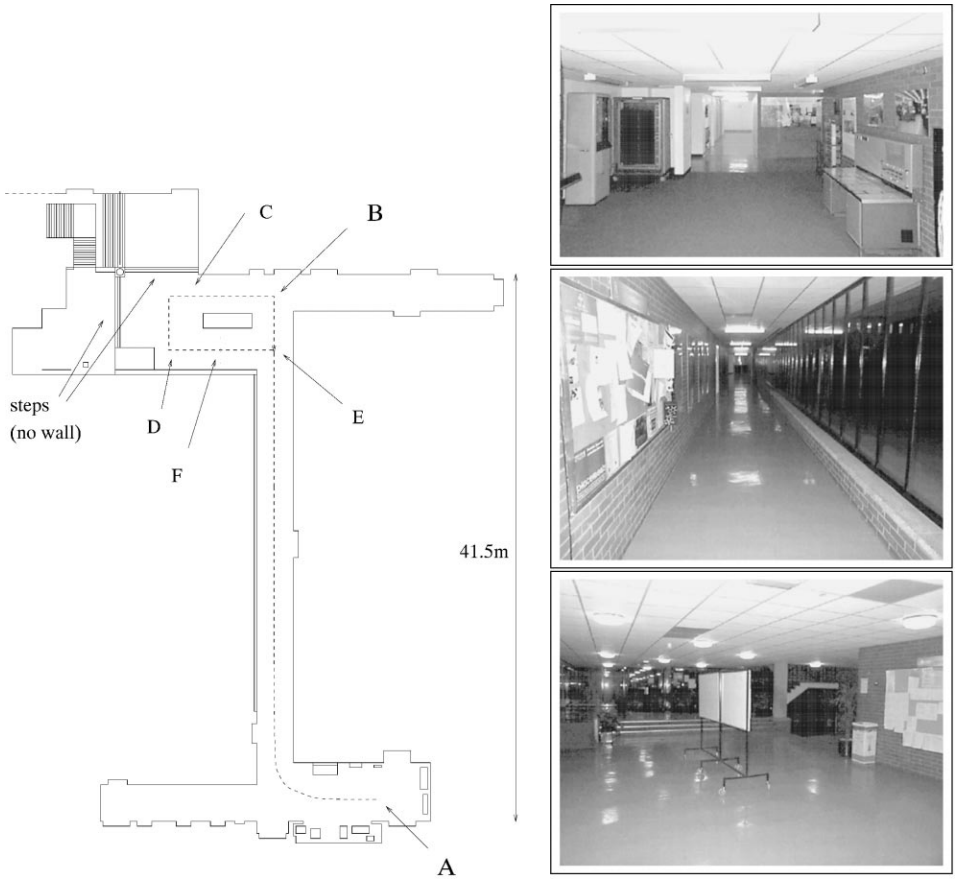

Fig. 15. A large-scale environment. The top photograph shows a view from location *A*, the middle shows a view from *E* looking down the corridor, the bottom photograph also shows a view from *E* but now looking towards *C*.

robot to follow). On completion of one loop (at landmark *A*) the robot was turned by approximately 90° anti-clockwise, and a "forward" command issued. On travelling in this direction, a new landmark was generated (i.e. *E*) and on reaching the opposite wall a match was made with landmark *B* (i.e. the robot was within the perceptual area of *B* and was "close enough" to the position stored for *B*). The robot was then guided around to landmark *C*, turned by 90° and again given the forward command. This resulted in a new landmark, *D*, a match with landmark *E* and a match with landmark *F* on the opposite wall.

4.5.2. Route following

The accessibility of all landmarks on the periphery was tested by presenting the robot with the following sequence:

1. plan and execute a path from *J* to *A*;
2. plan and execute a path from *A* to *K*;
3. plan and execute a path from *K* to *L*;
4. plan and execute a path from *L* to *J*.

This sequence was then repeated in reverse. The full sequence of forward and reverse was repeated five times. In all cases traversal was successful.

From landmark *J*, the robot was then presented with the following sequence:

1. plan and execute a path from *J* to *H*;
2. plan and execute a path from *H* to *G*;
3. plan and execute a path from *G* to *K*;
4. plan and execute a path from *K* to *D*;
5. plan and execute a path from *D* to *J*.

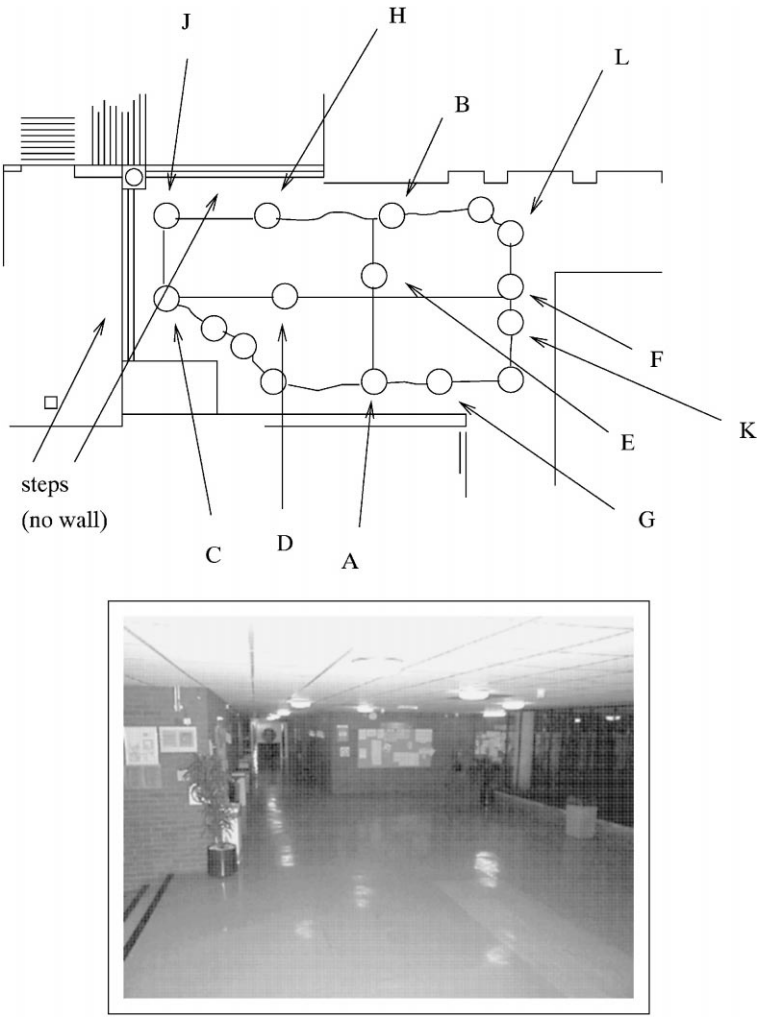

Fig. 16. An open environment. The photograph was taken from location *J*.

Step 2 ensures that a path via *E* is traversed. Steps 4–5 ensure that a path via *D* is traversed (*K*–*J* cannot be used since the path via *L* is marginally shorter). This sequence was then reversed. As with the previous sequence, this was repeated five times. Again, in all cases traversal was successful.

4.6. Conclusions drawn from experimental results

Repeated experiments were conducted in a wide range of different, unmodified environments. Environments included a (relatively small) robotics laboratory, which contained furniture and boxes, an unmodified workshop furnished with lathes, milling machines and toolboxes, various corridors of several hundred metres length, open spaces which were sparsely furnished, but contained a high number of people, and an environment which comprised both corridor areas and open spaces.

In all but the mechanical workshop, the robot was able to traverse several specified routes repeatedly (usually, routes were traversed 10 times) with a success rate of 100%.

The experiments in the workshop, however, reveal one weaknesses of the approach: whilst the navigation

system proved to be very robust with respect to temporary disturbances of the earth's magnetic field (which happen, for instance, when the robot passes a metal pillar), navigation becomes very hard, if not impossible, if magnetic disturbances are large and persist over long periods of time. The workshop environment was full of metal machinery that caused widely fluctuating compass readings, resulting in two out of the 34 perceptual landmarks in the workshop environment having very high failure rates in centring (between 20 and 80%). However, even in a very difficult environment such as the mechanical workshop, the robot was able to centre on 32 of the 34 landmarks with a success rate of 100%.

One approach to solving the problem of magnetic disturbance would be to apply some form of damping to the compass mechanism so that smoothing is applied to any sudden change in the compass reading. Another approach would be to use the environment itself to find an orientation for the input vector. Kurz [8], e.g. uses a method whereby he performs a virtual rotation of the input vector towards the area that looks most occupied according to the sonar sensors, i.e. where the sensors measure the shortest distance.

5. Discussion and conclusions

5.1. Related work

There are several robot navigation mechanisms that share certain properties with the system proposed in this paper.

Mataric [9] uses a topological representation of the environment, which is built by the robot using wall following. The types of landmark that can be detected in this scheme are very simplistic, being restricted to left walls, right walls and corridors. In complex domains, simple features such as these may not be useful in categorising the environment. Since our mechanism is not reliant on such user defined landmarks, it is more suited to map building in complex real world situations. In addition, in using wall following, the type of environment that can be mapped is highly restricted. In our scheme, user guidance of the robot is possible, thus removing the restrictions that may be imposed in using any particular exploration strategy.

Kurz [8] uses a self-organising approach to landmark generation that is similar to that used in our navigation mechanism. A topological representation is formed using the landmarks generated by the system. Here, positional information is attached to each landmark, and corrected using a method incorporating an extended Kalman filter. As with our mechanism, a standard graph searching method is used to generate a path to the goal using the topological representation. However, in Kurz' scheme it is the stored and current estimated positional information that is used for calculation of the trajectory to be taken for traversal between the landmarks. This can be a problem where the distance between landmarks is great. In this instance, inaccuracy in calculation of the directional component may lead to the robot missing the successive landmark completely. In contrast, with our system positional information is used for disambiguation alone, traversal between landmarks is achieved through a combination of behaviours. Our approach, therefore, is much less dependent on accurate positional information.

Similarly, Zimmer [21] constructs a topological map using a self-organising approach to landmark generation. Here, an adaptive network is used in which each cell represents a "situation" (i.e. a number of sensor readings combined with positional information). New cells are added to the net when the distance between the current situation and its closest match in the network (calculated using the Euclidian norm) exceeds a pre-determined threshold. To reduce the effects of drift, the robot's positional information is corrected by continuous correlation with previously mapped areas of the environment. This system also exhibits the ability to operate in dynamic environments. Changes to the environment are incorporated into the map, and the robot is able to re-plan if a path to the goal becomes blocked. However, the experiments reported in this paper were conducted in small-scale laboratory environments. How Zimmer's system would perform in larger, more complex environments is an open question.

A view-based approach is detailed in Franz et al. [4]. Here, a graph representation is derived using visual information from selected positions within the environment (these positions being selected automatically by the system itself). In a similar manner to our own mechanism, centring at a landmark is achieved by

reducing the difference between the current perception and a stored snapshot. In contrast to our approach, however, this is the only mechanism used for traversal between the landmarks. A major drawback with the approach detailed in [4] is that no account is taken of perceptual aliasing (i.e. two or more distinct locations appearing identical). Erroneous entries will therefore corrupt the map whenever this common phenomenon occurs. In addition, landmarks have to be visible from one another. For long uneventful corridors this is inefficient.

A topological mapping scheme in which each node simply contains positional information is detailed by Yamauchi and Beer [20]. In this scheme, an *Adaptive Place Network* is used to determine the nodes of the robot's map. Each node is identified by its Cartesian co-ordinate (as determined by the robot's onboard odometry mechanism). During map building, behaviour-based control is used for exploration. As the robot moves through the environment, the distance to the nearest node is continuously calculated. If this distance is larger than a pre-determined threshold, then a new node is created containing the robot's current position. Otherwise, the robot is deemed to be at the location given by the nearest node. However, since it is the robot's odometry that is used to determine location, some action must be taken to counter the effects of drift. The method used is to return the robot periodically to a "home" location for re-calibration of its odometry. Re-calibration is achieved by constructing an evidence grid [10] at the "home" location. On return to that location after a period of exploration, a new evidence grid is constructed, and a transformation is performed to find the best match between the grids. This transformation is then applied to the robot's dead reckoning position estimate to determine the robot's actual position. This approach severely restricts the size of environment that can be mapped, since the time that can be allowed for exploration is limited; if the robot is too far from its "home" location on return (due to odometry drift) then there may be no correlation between the stored evidence grid and the one generated on return to home.

5.2. Conclusion

The goal of the experiments described in this paper was to build a reliable, robust robot navigation system, that would fulfil the following requirements:

- the robot was to navigate in unmodified environments;
- had to navigate in areas of several hundred square metres in size;
- had to cope with short-term changes (such as passing people and opening doors) and long-term changes (such as blocked passage ways);
- had to achieve these competencies without requiring any a priori information, such as CAD models of the environments, metric maps, etc.

Our earlier work had shown that topological maps of perceptual landmarks, evolved through self-organisation and robot–environment interaction, could achieve some of these requirements in the laboratory. The problem addressed in the experiments here was how to scale the navigation mechanism up to suit large-scale, dynamic environments.

We achieved our objectives in *large* environments by adding a number of enabling behaviours.

The *centring behaviour* ensured that the robot would move towards the centre of a perceptual landmark, using sensory perceptions alone to achieve this. The result of the centring behaviour was to compensate for any odometry errors that had accumulated on the journey between landmarks.

The *wall following behaviour* exploited the fundamental features of some parts of our environments, by using the walls to guide the robot towards the next landmark. This increased reliability markedly.

The *returning behaviour*, a modification of the centring behaviour, allowed the robot to retrace its steps after having failed to localise the next landmark.

The *link forgetting behaviour* ensured that the robot could incorporate long lasting changes to the environment into its map. This would allow the robot to modify its path planning strategy accordingly.

Enabling mechanisms used were the multiple coding of landmarks for junctions and open spaces, and the combination of local odometry information and sensory perception to address the problem of perceptual aliasing.

Apart from one environment, in which metal objects introduced a large error on the robot's flux gate compass, the robot was able to traverse all selected routes, in environments of widely different

perceptual appearance. The conclusion to draw from these experiments is, in our view, that it takes a combination of adaptive processes and enabling behaviours to build robots that can navigate reliably over large distances, using only naturally occurring landmarks.

Acknowledgements

We express our thanks to David Brée and the anonymous reviewers for helpful comments on earlier versions of this paper.

References

[1] H.E.J. Borenstein, L. Feng, Navigating Mobile Robots, A.K. Peters, Wellesley, MA, 1996.
[2] K. de Meyer, O. Lemon, U. Nehmzow, Multiple resolution mapping for efficient mobile robot navigation, in: TIMR–97 Towards Intelligent Mobile Robots, Technical Report Series UMCS-97-9-1, University of Manchester, 1997, ISSN 1361-6161, http://www.cs.man.ac.uk/csonly/cstechrep/titles97.html.
[3] T. Duckett, U. Nehmzow, A robust perception-based localisation method for a mobile robot, Technical Report Series UMCS-96-11-1, Department of Computer Science, University of Manchester, 1996.
[4] M.O. Franz, B. Schölkopf, H.A. Mallot, H.H. Bülthoff, Learning view graphs for robot navigation, Autonomous Robots 5 (1998) 111–125.
[5] L. Kleeman, Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead reckoning, in: Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France, 1992, pp. 2582–2587.
[6] T. Kohonen, Self-Organisation and Associative Memory, Springer, Berlin, 1988.
[7] D. Kortenkamp, T. Weymouth, Topological mapping for mobile robots using a combination of sonar and vision sensing, in: Proceedings of AAAI-94, Seattle, WA, 1994.
[8] A. Kurz, Constructing maps for mobile robot navigation based on ultrasonic range data, IEEE Transactions on Systems, Man, and Cybernetics 26 (2) (1996) 233–242.
[9] M.J. Mataric, Integration of representation into goal-driven behaviour-based robots, IEEE Transactions on Robotics and Automation 8 (3) (1992) 304–312.
[10] H. Moravec, A. Elfes, High resolution maps from wide angle sonar, in: Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, MO, 1985, pp. 116–121.
[11] U. Nehmzow, Experiments in competence acquisition for autonomous mobile robots, Ph.D. Thesis, Department of Artificial Intelligence, University of Edinburgh, 1992.
[12] U. Nehmzow, T. Smithers, Mapbuilding using self-organising networks, in: J.A. Meyer, S. Wilson (Eds.), From Animals to Animats, Proceedings of the SAB90, Paris, MIT Press, Cambridge, MA, 1991, pp. 152–159.
[13] U. Nehmzow, T. Smithers, J. Hallam, Location recognition in a mobile robot using self-organising feature maps, in: G. Schmidt (Ed.), Information Processing in Autonomous Mobile Robots, Springer, Berlin, 1991.
[14] C. Owen, Landmarks, topological maps and robot navigation, Master's Thesis, Department of Computer Science, University of Manchester, 1995.
[15] C. Owen, Map-building and map-interpretation mechanisms for a mobile robot, Ph.D. Thesis, Department of Computer Science, University of Manchester, 2000.
[16] C. Owen, U. Nehmzow, Route learning in mobile robots through self-organisation, in: Euromicro Workshop on Advanced Mobile Robots, IEEE Computer Society, 1996, ISBN 0-8186-7695-7.
[17] C. Owen, U. Nehmzow, Landmark-based navigation for a mobile robot, in: From Animals to Animats, Proceedings of the SAB98, Zurich, Switzerland, MIT Press, Cambridge, MA, 1998.
[18] C. Owen, U. Nehmzow, Map interpreta- tion in dynamic environments, in: Proceedings of the Fifth International Workshop on Advanced Motion Control, IEEE Press, New York, 1998, pp. 340-345, ISBN 0-7803-4484-7.
[19] D.L. Reilly, L.N. Cooper, C. Erlbaum, A neural model for category learning, Biological Cybernetics 45 (1982) 35–41.
[20] B. Yamauchi, R. Beer, Spatial learning for navigation in dynamic environments, IEEE Transactions on Systems, Man, and Cybernetics 26 (3) (1996) 496–505.
[21] U.R. Zimmer, Robust world modelling and navigation in a real world, Neurocomputing 13 (2–4) (1996) 247–260.

Ulrich Nehmzow is a Lecturer in Artificial Intelligence and Robotics at the Department of Computer Science at the University of Manchester, leading its Robotics Research Group.

He graduated in 1988 from the Technical University of Munich in Electrical Engineering and Information Science (Dipl. Ing.), and obtained a Ph.D. in Artificial Intelligence from the University of Edinburgh in 1992. Following appointments as Research Associate at the Department of Artificial Intelligence, University of Edinburgh, and postdoctoral Research Associate at the Department of Psychology, also at Edinburgh, he became a Lecturer at Manchester in 1994.

He is a member of the Institution of Electrical Engineers, member of the IEE Northwest Center Committee, member of the Society for Artificial Intelligence and Simulation of Behaviour, and has worked as Visting Research Scientist at Carnegie Mellon University and the University of Bremen. In 1999 he was awarded a Royal Society/STA fellowship for a 7-month sabbatical at the Electrotechnical Laboratory in Tsukuba, Japan.

His research focusses on autonomous mobile robotics, robot learning, navigation, artificial neural networks, robot simulation and novelty detection

Carl Owen studied computing at Nottingham Trent University. He obtained his B.Sc. in 1994, graduating with first class honours. His final year project involved the mathemtical modelling of biological structures using the formalism of L-systems.

He obtained an M.Sc. in Advanced Computer Science from Manchester University in 1995, the project here involving the use of a self-organising controller that would allow a mobile robot to learn to navigate along a given route.

In 2000 he completed his Ph.D. studies at Manchester University.