ISSN 1361 - 6161

THE UNIVERSITY *of* MANCHESTER

Computer Science
University of Manchester

A ROBUST, PERCEPTION–BASED LOCALISATION METHOD FOR A MOBILE ROBOT

T. Duckett and U. Nehmzow

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-96-11-1

A ROBUST, PERCEPTION–BASED LOCALISATION METHOD FOR A MOBILE ROBOT *

T.Duckett and U. Nehmzow

Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
t.duckett@cs.man.ac.uk, u.nehmzow@cs.man.ac.uk

21/10/96

*Refereed by: R. Dixon

Abstract

This paper addresses the problem of localisation in autonomous mobile robots, which is the task of identifying places that have been visited before by the robot. In particular, this paper is concerned with the more general problem of relocalisation, where the robot is unable to localise using past experience (in other words, the robot is "lost"). A perception based localisation algorithm is presented, which operates independently of any global reference frame. The method is resilient to the problem of perceptual aliasing (i.e., perceptual identity of distinct locations), and is capable of localising even in environments where no single place has a unique perceptual signature.

During an exploration phase, the robot builds a map of its environment, using a self–organising neural network to cluster its perceptual space. The robot is then moved to a randomly chosen position, where it will attempt to localise. By actively exploring, and accumulating evidence through the use of relative odometry between local landmarks, the robot is able to determine its location with respect to perceptual landmarks very quickly.

1 Introduction

In the field of mobile robot navigation, (self-) localisation refers to the task of establishing one's position with respect to the known world. In particular, this paper is concerned with the worst case of relocalising after becoming lost through whatever circumstance. This is equivalent to Engelson's "kidnapped robot problem" [6], where the robot is picked up and moved to an arbitrary location while being deprived of any sensory input. Thus, no *a priori* estimate of position is available to the robot.

People inevitably observe the world differently to robots, a problem described as perceptual discrepancy by Nehmzow and Mitchell [17]. We therefore believe that the robot itself should acquire its knowledge of the world, e.g., avoiding the use of pre-installed maps wherever possible. As we are interested in the use of autonomous mobile robots in unmodified environments, we also reject the use of artificial beacons or other external devices for position evaluation. Furthermore, the accumulated drift error caused by wheel slippage means that dead reckoning alone cannot be used for robust localisation. Thus, the problem becomes one of correlating past and present impressions of the world, or in other words, recognising places you've seen before based on your most recent perceptions.

The task of localisation is made non–trivial by the fact that many places may look the same, i.e., share the same "perceptual signature", a characteristic of navigation known as perceptual aliasing. In addition, the sensory information available to the robot is prone to noise, and may often be inconsistent as well as imprecise.

Existing methods for localisation may be roughly categorised according to the approach used to resolve perceptual aliasing (see section 2). Some authors attempt to disambiguate similar looking locations by adding history information based on previous sensory experience to the perceptual signature. Others use a prior estimate of position, usually obtained by dead reckoning, which is then corrected by matching current sensory perceptions with some internal world model. However, in reasonably complex environments neither of these approaches is sufficient for solving the general relocalisation problem. The work presented in this paper represents an attempt to find a more general method of localisation.

The robot begins by building a map of its environment, using a self–organising neural network (ART2) to cluster its perceptual space. The robot is then moved to a random position in its environment, its sensors being disabled during that move. An active exploration phase follows, where the robot accumulates evidence based on relative odometry between perceptually distinctive locations. Hence, previous sensory experience is used to choose between competing hypotheses of the robot's possible location. The resulting position estimates are corrected by reference to the previously acquired world model.

Both the map building and localisation competences operate independently of the exploration strategy used. In this paper, performance is demonstrated using two different exploration behaviours (contour following and wandering), both having been acquired autonomously by the robot using instinct rules (see Nehmzow [13] for details). The system as a whole is composed of a hierarchy of behaviours (see figure 1), each being resilient to a significant degree of error in the preceding levels. Thus, the localisation system

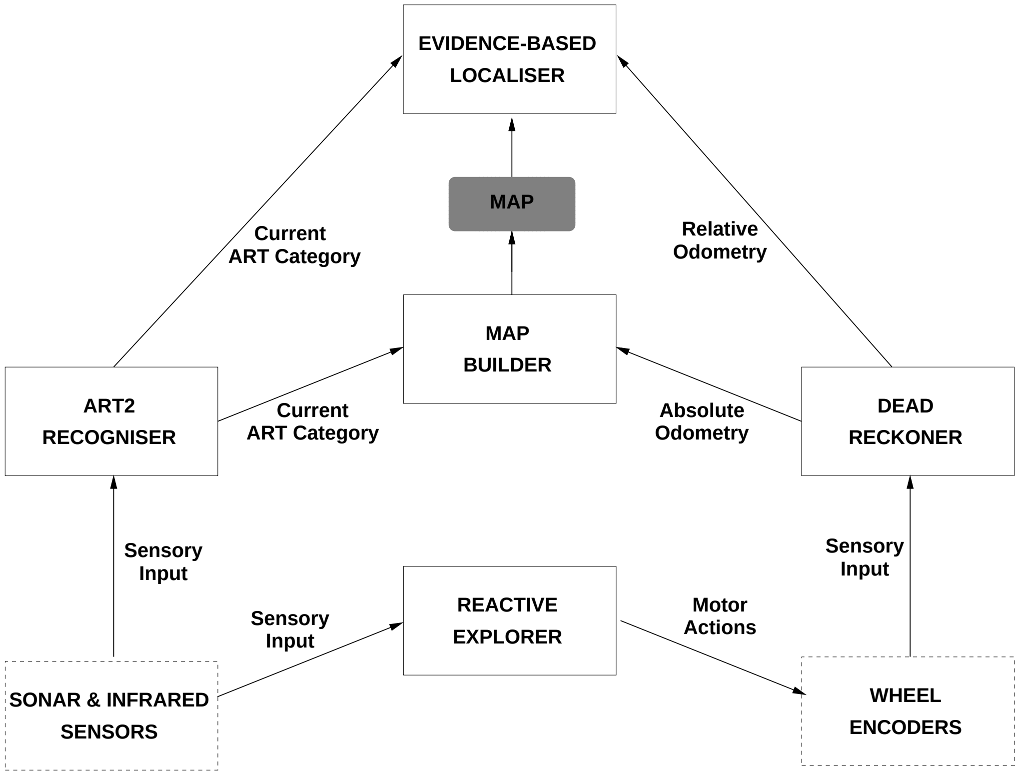

Figure 1: *Current System Architecture.* The solid-edged boxes denote behavioural modules, and arrows the dependencies between different parts of the system. The shaded box denotes the representation used for the map, and the dashed boxes show hardware components. Possible extensions to the basic architecture are discussed later in section 7.3.

is highly robust, and has localised successfully in hundreds of laboratory trials, both on a real robot and using a simulator.

The rest of this paper is structured as follows; Section 2 gives an overview of related work in the field. Sections 3 to 5 detail the perceptual clustering, map building and localisation components respectively. Experimental results are given in section 6, while section 7 presents the final conclusions including the contribution made by this work, problems found and directions for future research.

2 Related Work

Existing work relating to mobile robot localisation could be partitioned along various different dimensions. The current estimate of position may be explicit, being anchored to some global reference frame [10] [22] [23], or implicit [18] [20], e.g., Owen [18] directly associates sensor patterns with motor actions, obviating the need for any symbolic representation of position. Similarly, the robot may or may not possess an explicit world model. Lee [11] provides a useful taxonomy of world models, distinguishing Recognisable Locations, Topological Maps, Metric Topological Maps and Full Metric Maps.

Perhaps the most useful categorisation can be made according to the approach used in handling perceptual aliasing. Some authors make no provision for perceptual ambiguity, assuming a one-to-one mapping between (unique) perceptual signatures and physical locations. Others attempt to distinguish similar looking locations by using more and more sensory information from an increasing number of sensor modalities. Bühlmeier *et al* [2] have successfully used this method.

However, increasing the robot's perceptual resolution will not solve the more general relocalisation problem because the absence of perceptual aliasing can never be assumed with certainty in complex environments. Navigation strategies based on this assumption are bound to fail sooner or later. Furthermore, some degree of perceptual aliasing is actually desirable, in that it provides a generalisation on perception.

Another approach for dealing with perceptual ambiguity is to incorporate history of local sensor sequences into the recognition of locations. Nehmzow and Smithers [15] combine past and present sensorimotor experience of the robot in the input to a self–organising feature map. Similarly, Tani and Fukumara [20] add previous sensor patterns to the input field of a neural network controller.

A disadvantage of this approach is that the identical set of previously stored locations must be revisited for matching of history information to be possible. In complex environments, this can only be achieved by restricting the navigational behaviour of the robot, often to following fixed paths. Nehmzow *et al* [16] showed that a range of temporal horizons or "time windows" of history is required to localise in an environment containing perceptual aliasing, i.e., one fixed length of history was shown to be insufficient. In general, this approach is limited by the upper limit on history stored, and does not generalise to cover arbitrarily large areas of perceptual ambiguity in the robot's environment. (For example, in a large office environment such as the Computer Science Department at Manchester University, whole regions of the building may look very similar.) Also, these methods are not robust to sensor noise; one freak perception will affect the next n inputs to the system, where n is the number of previous perceptions taken into account for localisation.

A further approach effectively uses dead reckoning to disambiguate similar looking locations. The estimate of position provided by dead reckoning is corrected by matching observed features of the environment against an internal world model [1] [5] [8] [9] [10] [22] [23]. For example, Kuipers and Byun [9] move the robot itself towards locally distinctive places as a means of correcting the drift error due to odometry. While the reliance on dead reckoning makes this approach unsuitable for solving the general relocalisation problem when the robot is lost, these methods do contain some useful techniques for map building and localisation.

The maps used in these localisation methods are often graph–based, using topologically connected "place" nodes. Metric information, corresponding to the relative translational and rotational displacement of the robot between places, may be added [10] [22] [23]. Position updates may be continuous [10] or episodic [22]. Some methods are capable of dealing with dynamic environments [22] [23], and even of distinguishing transient changes such as moving people from lasting changes in the environment [22].

The research presented in this paper incorporates aspects of both of the latter approaches to resolving perceptual aliasing. Evidence based on past sensory experience is used to mediate between competing position estimates, which are corrected by matching current perceptions to an autonomously acquired world model. The map used is similar to the graph–based models described above (detailed in section 4). An explicit world model was used partly because of the growing consensus over the need for some form of representation, even amongst strongly behaviour–based researchers, e.g, [12], and partly because this particular representation made possible the development of the algorithm for evidence accumulation (see section 5).

The work is also related to hippocampal models of localisation [19]. The map used consists of a discrete set of distinct locations; these are analogous to place cells. The localisation algorithm is essentially a competitive learning scheme; the location in the map with the highest "activation" or confidence level provides the winning position estimate.

3 Perceptual Clustering

The first stage of the problem is for the robot to recognise distinct places in its perceptual space. This is complicated by the problems of using sensors in the real world. For example, the robot will often obtain different sensor readings when revisiting a previously encountered location. Individual sensory perceptions may therefore be inconsistent, imprecise and unreliable. Hence, the robot should be capable of generalising in order to extract the salient features in a given situation, without being distracted by the finer detail of individual sensor patterns.

In accordance with the general principle of autonomous learning by the robot, a self–organising classification system is used. The neural network architecture ART2 [4] is responsible for clustering input patterns into distinct classes or categories, such that similar patterns are grouped into the same class and dissimilar patterns are grouped into separate classes. The stored class definitions may be said to correspond to prototypes or templates for matching with current perceptions of the world.

The motivation for choosing this approach is that clustering the robot's perceptions autonomously using a self–organising classifier helps to avoid the "correspondence problem" of matching individual environmental features against an internal world model. There is no attempt to recognise specific objects in the robot's environment, rather the raw sensor readings are grouped according to their similarity. This means that the robot's perceptual groupings (ART classifications) may bear no direct translation to obvious human categorisations of environmental features (e.g., "corner", "wall", "box", etc). Suitable perceptual landmarks emerge rather than being defined arbitrarily by a human designer.

ART (Adaptive Resonance Theory) is not the only methodology that could be used for the localisation task. Other possible clustering techniques would include Self–Organising Feature Maps [10] [15] [18], Restricted Coulomb Energy networks [10], Growing Cell Structures [23], Vector Quantization networks, etc. The specific advantages and disadvantages of using ART are discussed in the next section.

In the wider context of the map building and localisation competences, which are presented later in the paper, the ART network is effectively used as a "black box" for classifying sensor patterns. Readers who are not interested in the details of the ART architecture might wish to skip over some of the following material.

3.1 Characteristics of ART

Some of the important characteristics of ART are listed as follows, explaining the motivation for choosing this particular classification strategy.

- *Unsupervised Learning.* Self–organisation means that suitable prototypes emerge rather than being hand–crafted by the designer.
- *Real–Time Learning.* No off–line processing is required.
- *Lifelong Learning.* Learning is continuous during normal operation, so there are no separate training and testing phases. (This characteristic has not yet been exploited in this work.)

- *Solution to the "Stability–Plasticity Dilemma".* The network can learn new information without forgetting old information, simply by adding more prototype nodes to the network. In other competitive learning schemes, e.g., Kohonen, the size of the network and therefore the amount of storable information must be fixed in advance.
- *Variable Sensitivity to Perceptual Detail.* A pre–specified variance parameter known as the "vigilance" determines the size of the clusters. A high value results in fine categories, a low value in coarse categories.
- *Closed Classification Boundaries.* An input pattern cannot be classified in a particular category if the similarity falls below the vigilance threshold. Thus, ART can detect whether an input pattern has been "seen" before or not. Other competitive networks, e.g., Vector Quantization, Kohonen, etc., will always match with the closest node, even if the input pattern bears little resemblance to any of the stored patterns. This has the advantage that there is a clear criterion for distinguishing familiar from unfamiliar input patterns, while losing the ability of competitive networks to "second guess" an appropriate classification based on noisy input data.
- *Self-Scaling Property.* This prevents any pattern that is a subset of another from being classified in the same category. Thus, patterns which share common features but are in different classes can still be distinguished. Grossberg [7] calls this "the discovery of critical features in a context sensitive manner".

However, ART does have some disadvantages too. The general complexity and reliance on details of architecture have been criticised, and there are a large number of parameters or "magic numbers" which must be determined experimentally by the designer. The solution to the "stability-plasticity dilemma" of adding more nodes means that ART may continue to come up with new prototypes even if the robot stays within a previously explored part of the environment, e.g., due to sensor noise. Another problem found was over–training, where new categorisations occurred in physical locations which had previously been associated with a different ART category. This is also discussed in more detail in section 3.4.

3.2 The ART Information Processing Model

The basic ART architecture consists of two fully connected layers of units, a feature layer (F1) which receives the sensory input, and a category layer (F2) where the units correspond to the perceptual clusters or prototypes (see figure 2). There are two sets of weights between the layers, corresponding to feedforward and feedback connections. A "winner-takes-all" criterion is used during the feedforward phase, and a similarity criterion is used to accept or reject the resulting categorisation in the feedback phase.

When a pattern is presented to the network, the input is compared to each of the existing prototypes (through the feedforward weights) to determine a winning node. If the similarity between the input pattern and the winning node (through the feedback weights) exceeds the vigilance threshold, then adaptive learning is activated and the stored pattern is modified to be more similar to the input pattern (the learning method depends on the particular ART network used - see next section). Otherwise a "reset" occurs, where the current winner is disabled, and the network searches for another node to match the input pattern. If none of the stored prototypes is similar enough to the given input, then a new prototype will be created corresponding to the input pattern.

In earlier clustering experiments, the ART1 network [7] was tried, but was found to be unreliable because of the requirement of binary inputs. The individual sensor readings had to be coarse coded into "bins" (i.e., '0' or '1') according to a threshold level. It was found that sensor noise around the threshold and variations in the exploration behaviour of the robot produced too many inconsistent input patterns, resulting in misclassifications and spurious prototypes being created. The ART2 network was therefore implemented, which is capable of handling continuous valued inputs.

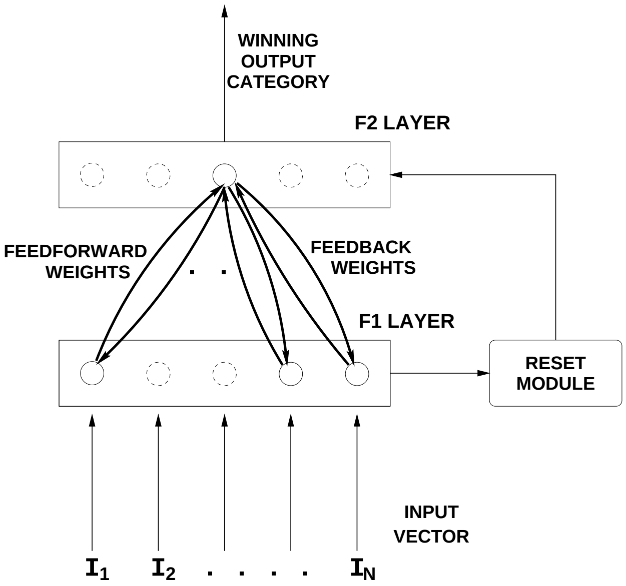

Figure 2: *Basic ART Architecture.* The units in the feature layer (F1) are completely connected to each of the units in the category layer (F2), through two separate sets of weights. Only a few of the feedforward and feedback connections are shown for clarity.

3.3 ART2 Implementation

The ART2 architecture is a generalisation of ART1, being capable of learning and recognising continuous valued input patterns. The basic connectivity is the same as in ART1, except that the units in the feature layer (F1) are each constructed from a subnetwork of six nodes (see figure 3). Each subnetwork act as a buffer between the input signal and the feedback from the category layer (F2), being used to normalise and combine the two signals for comparison by the reset module. In addition to the basic architecture, an extra pre-processing layer (F0), consisting of a subnetwork of four nodes per unit, was added to further reduce noise and enhance the contrast of the input patterns, as suggested by Carpenter and Grossberg [4]. Full details of the architecture and equations describing the dynamics of ART2 may be found in Appendix A.

In ART2, adaptive learning is carried out by moving the stored weight vectors in the direction of the input vector (the magnitude of vectors is ignored due to vector normalisation). The "fast" mode of learning was implemented, where the network is allowed to completely stabilise after each training pattern is presented. This means that that the weight vectors are moved to the fullest extent possible, allowing locations to be remembered by the robot after a single visit. The Runge-Kutta method [3] was used to solve the differential equations used for carrying out the weight adjustment.

The input vector to the ART2 network was taken from either the 16 infra-red or 16 sonar sensors mounted on the turret of the mobile robot "FortyTwo", depending on the experiment or simulation being carried out (see section 6.1). The sensor patterns were always presented in the same orientation, regardless of the direction travelled by the robot. This was done by keeping the robot's turret stationary to provide an artificial compass sense (see section 6.1 for more details).

The implementation was used successfully to classify the robot's perceptual space consistently between discovering and revisiting the same locations. Figure 4 shows the results of a simulation [1] where ART2 was run in parallel with a wall following behaviour. The robot's physical space is divided into regions according to the ART classification category (these areas are termed "perceptual regions" in the

[1] In the simulations carried out, sensors and actuators were modelled using the Nomadic Robot Simulator package.

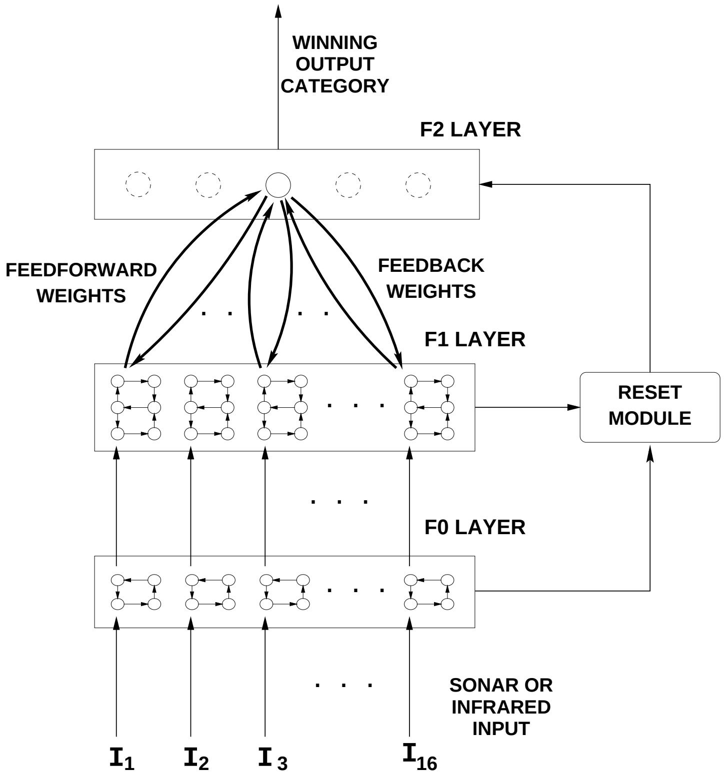

Figure 3: *ART2 Architecture Used For Localisation.* Each F1 unit consists of a sub-network of six nodes, and each F0 unit consists of a sub-network of four nodes. Full details may be found in Appendix A. The F1 and F2 layers are completely connected; only a few of the connections are shown for clarity.

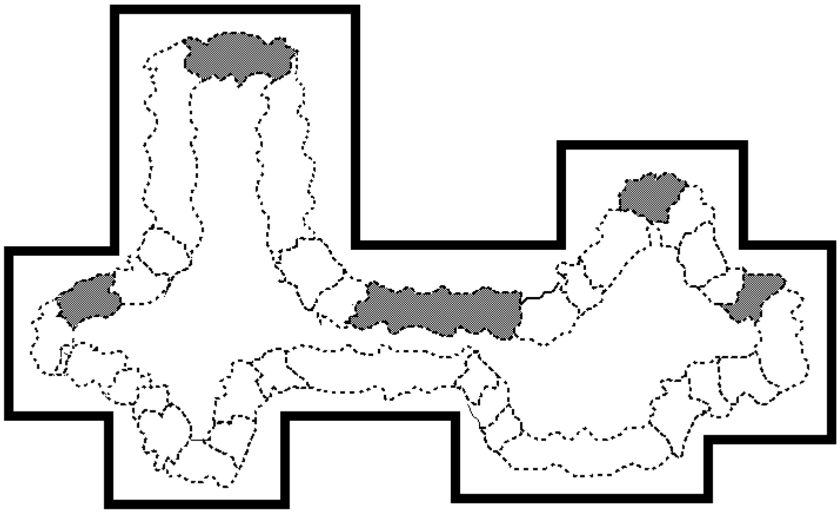

Figure 4: *ART2 Classifications During Wall Following Simulation.* As the robot followed the walls of a room, the physical space covered was divided into regions according to the ART category recorded. An example of perceptual aliasing is shown by the shaded regions; these all share the same ART classification.

rest of the paper). An example of perceptual aliasing is shown by the shaded areas; these regions all share the same ART category.

3.4 Problems

In ART networks, a novel perception always results in the creation of a new prototype. Thus, noisy or "rogue" input patterns can result in spurious ART categories being created as well as misclassification errors. In the real world using real sensors, no classification method will probably ever give completely error free performance. However, the ART2 implementation rarely suffered from either of these problems, and performance was sufficiently good for the localisation algorithm to succeed every time in the experiments conducted (see section 6).

In addition, "over–training" was found to be a problem during prolonged use of the ART2 network under normal operation, e.g., after 5 or 6 laps of a room by the wall follower, where new classifications started to occur which were inconsistent with the results in previous laps. This was because the prototype vectors continue to be adjusted during lifelong learning, as the stored patterns are continuously moved closer to the input patterns presented. It was found that new prototypes were occasionally created to fill the "holes" which appeared between the existing, previously overlapping clusters moved by adaptive learning.

This problem was avoided in the localisation system presented here because training was switched off after map building had been carried out, and the patterns remained fixed during localisation. For example, during wall following, training was finished once a complete circuit of the robot's enclosure had been detected by dead reckoning. In future work, the problem might be avoided by disabling the training on individual prototype nodes after a given number of updates, while retaining the ability to train additional nodes with new patterns.

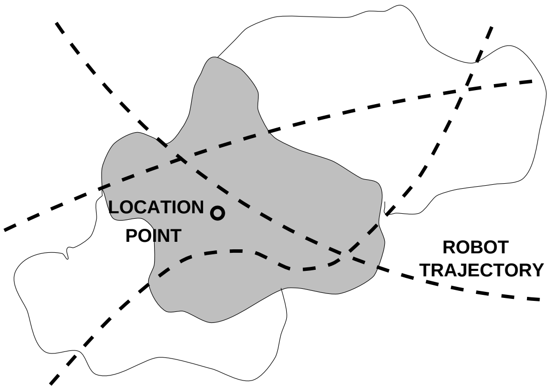

Figure 5: *Creation of Location Points.* As the robot moves through a region corresponding to a particular ART classification (shaded), the x and y co-ordinates are averaged continuously to give the position of the corresponding location point. Adapted from Kurz [10], figure 7.

4 Map Building

After perceptual clustering has been carried out, the next stage is to build a map of the locations visited during exploration. The map created contains stored "locations", where a location is an ART category associated with an (x, y) coordinate obtained by dead reckoning. New locations are entered into the map either when a new ART category is perceived, or the robot has moved by more than a distance threshold D. [2]

The map represents a clustering of the Cartesian space within the regions defined by the ART classifier, consisting of a discrete set of stored points. The stored locations could therefore be said to correspond to "place prototypes". The representation accommodates perceptual aliasing because there may be a one-to-many relationship between perceptual signatures (ART categories) and stored locations in the map.

In Lee's taxonomy [11], the world model might be described as "Recognisable Locations", except that metric information is also represented. If the map were to be incorporated into a complete navigation system, it would probably be necessary to store topological links as well for path planning, etc. (The version of the localisation algorithm presented later uses the metric information in the map to determine the relative proximity of locations, although topology could also be used for this purpose.)

4.1 Method

The x and y co-ordinates of the robot are averaged continuously as the robot moves across a perceptual region corresponding to a particular ART category, as shown in figure 5. Whenever the robot moves into a different perceptual region, a new location point is created. This point will keep moving towards the approximate centre of the cluster due to the averaging process, until a new cluster is formed. This is similar to the map building system described by Kurz [10].

In addition, when the displacement of the robot from the current (nearest) point location exceeds the distance threshold D, a new point location is also created. This leads to multiple point locations within large perceptual regions (see figure 6). Also, this handles the problem of perceptual aliasing, because different places which share the same perceptual signature will be represented by different points in the map, provided they are more than distance D apart. Otherwise, for example, averaging the x and y coordinates of all the shaded regions in figure 4 would result in a point somewhere in the middle of the

[2] In our experiments, $D = 25cm$ was chosen (see also Appendix B).

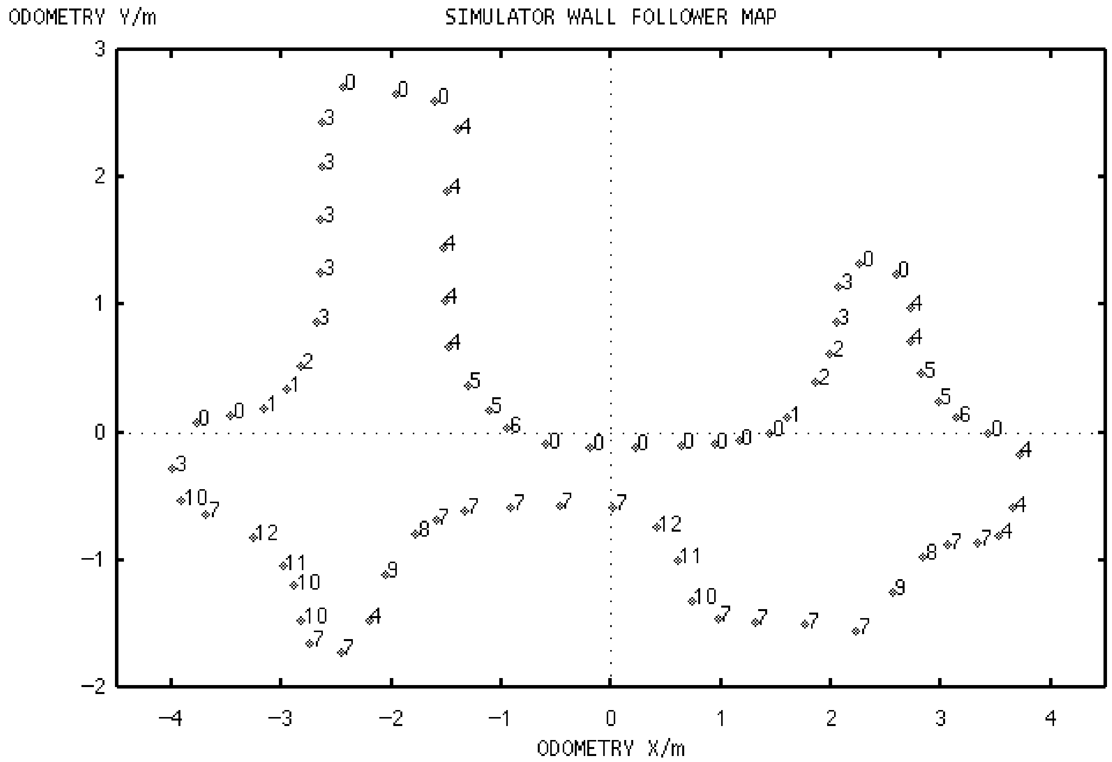

Figure 6: *Location Points Created During Wall Following.* The location points plotted here correspond to the perceptual regions shown in figure 4. The numbers shown correspond to the ART2 classifications, e.g, the points numbered '0' lie within the shaded areas shown previously.

room. This would be an incorrect mapping, as this point would correspond to none of the individual places from which it had been derived.

4.2 Coping with Odometry Drift

A major weakness of this part of the work is the reliance on global odometry for supplying the metric component of the map. (This problem does not affect the localisation algorithm as that only uses short distance odometry between local landmarks.) While the averaging process will smooth out local variations in odometry, the map as a whole will be subject to global drift errors. This was not a problem in the experiments conducted here as only relatively short routes were travelled, e.g., with the wall follower, only one lap of a room was used to build the map. However, in large, complex environments this could result in matching with the wrong "nearest" point during the map building phase.

This problem could be eliminated by combining the map building and localisation phases, or in other words, implementing lifelong learning. Thus, the map would be used to correct the current position estimate, which is used to build the map itself! Referring back to figure 1, this would effectively mean creating a link back from the localiser to the dead reckoner, applying appropriate corrections to the absolute odometry values based on the results of the localisation algorithm. This issue is discussed further in the conclusions. See also Kurz [10] and Zimmer [23] for some interesting work on this problem.

5 Localisation

The basic principle of the localisation method is similar to that of ART and other competitive learning schemes; the location point in the map with the highest "activation" or confidence level is selected as the winner. Thus, the location points are analogous to the "place cells" found in hippocampal models of localisation, e.g., see Recce and Harris [19]. The confidence levels are adjusted by accumulating evidence based on relative odometry between locations. The algorithm works by comparing old and new information at each iteration, or in other words, by considering the *changes* in the moving robot's perceptions over time.

5.1 Algorithm

Possible locations are stored as a list of hypotheses in working memory, each with an associated confidence level. Whenever the robot perceives a new location, either by recognising a change in the current ART category or a change in odometry bigger than the maximum distance between the stored location points, another set of possible locations is created. (The maximum distance between the stored locations will be $2D$ because the averaging process used in the map building means that the robot should be within the distance threshold D of the nearest point on either side.) A matching procedure is then used to combine both sources of evidence to produce an updated list of hypotheses. The algorithm is given as follows, and explained in more detail below.

0. *Initialisation.* Formulate set of hypotheses, $H = \{h_0, ..., h_N\}$, consisting of location points which match the current ART category. Initialise confidence levels: $\forall h_i \in H$, assign $conf(h_i) = 1$

1. Wait until ART category changes or robot has moved distance $2D$, where D is the distance threshold used during map building.

2. For each hypothesis h_i add change in odometry $(\triangle x, \triangle y)$ to the coordinates of h_i, (x_{h_i}, y_{h_i}).

3. Generate second set of candidate hypotheses, $H' = \{h'_0, ..., h'_N\}$.

4. For each h_i, find the nearest h'_j, recording distance apart, d_{h_i}.

5. Accumulate evidence, given distance threshold T, minimum confidence level MIN, gain factor $GAIN > 1$ and decay factor $0 < DECAY < 1$:
   $\forall h_i \in H$

   if $d_{h_i} < T$ then

   replace (x_{h_i}, y_{h_i}) with $(x_{h'_j}, y_{h'_j})$ from the matching h'_j

   let $conf(h_i) = conf(h_i) \times GAIN$

   else

   let $conf(h_i) = conf(h_i) \times DECAY$

   if $conf(h_i) < MIN$ then delete h_i.

6. Delete any duplicates in H, preserving the one with the highest confidence level.

7. Add all remaining h'_j from H' which are not already contained in H to H, assigning $conf(h'_j) = 1$.

8. Repeat from step 1.

In step 2, the existing set of location estimates are updated by adding on the change in position recorded by relative odometry. A new set of candidate hypotheses is then enumerated by selecting all possible location points from the map which match the current ART category. A search procedure follows, where each of the existing hypotheses is matched to its nearest neighbour in the new candidate set.

Step 5 then uses a threshold criteria to determine whether each of the matched hypotheses should be increased or decreased in confidence. This means that if the matched position estimates are sufficiently close to each other, then this is judged to be a piece of evidence in favour of this particular hypothesis. The confidence level is therefore raised by a gain term, and the position estimate from the old hypothesis is replaced by the new value. Thus, good position estimates are continually corrected on the basis of perception.

Conversely, if the match distance exceeds the threshold, then the associated confidence level is scaled down by a decay term, and the position estimate is left uncorrected, i.e, it is not replaced by its nearest neighbour in the candidate set. Hypotheses which fall below a certain confidence level are rejected and pruned from the list of possible locations, eliminating the bad hypotheses quickly and minimising the search space. Duplicates created by the matching process are deleted here too.

Any unmatched location points remaining in the candidate set are also added to the current list of hypotheses, and are assigned an initial confidence value. This will be the case for all of the candidate hypotheses on the first iteration, as the hypothesis set will initially be empty. (Initialisation has been included as step 0 here for clarity, although this is actually unnecessary.) All possible locations with the current perceptual signature are always considered, so that the algorithm can cope with unexpected, arbitrary changes of position.

Eventually, one of the hypotheses emerges as a clear winner. For example, in the experiments conducted on a real robot, this took an average of 7 iterations through the algorithm, in a mean time of 27 seconds with the robot travelling at 0.10 ms^{-1} (see section 6 for full details). If the robot becomes lost again, confidence in this particular hypothesis gradually decays as a new winner emerges.

5.2 Performance

The current version of the algorithm localises to the nearest stored location point. Thus, the accuracy of the position estimate depends on the resolution of the map, which in turn depends on the distance threshold D used in the map building process. After localisation has been achieved, the error between the actual and estimated position should therefore vary between zero and D (see next section). In future work, the localisation error could be smoothed out by merging successive position estimates, e.g., using the Extended Kalman Filter [10] [11], rather than by simply replacing an existing estimate with one taken directly from the map.

The results detailed in the next section also show that the algorithm is robust, being capable of localising even with extreme ART classification errors and odometric drift errors.

6 Results

In this section, details of the experimental platform used are provided. A set of experiments [3] are described where the localisation method was tested on a real robot. In addition, a series of simulations were conducted, so that the results obtained in a "perfect world" could be compared to others where different types of error could be deliberately introduced and carefully controlled.

6.1 Platform

The experiments were conducted on "FortyTwo", a Nomad 200 mobile robot, and the simulations on the Nomadic Robot Simulator, which is the corresponding software development package. The robot is supplied with coarse 360 degree sensing, consisting of both 16 sonar and 16 infra-red sensors. These sensors are mounted on a turret, which can be rotated separately from the base of the robot (see figure 7). The robot has three separate motors for translational, rotational and turret motion.

In previous work, e.g., [2] [18], the rotational and turret motors have been controlled in unison, so that the sensors always point in the same direction relative to the heading of the robot. Here, however,

[3] In this section, the term *experiment* is used to refer to the tests carried out on a real robot, and the term *simulation* is used to refer to the tests carried out on a simulated software robot.

Figure 7: *The Manchester 'FortyTwo'.* The robot's sonar and infrared sensors are mounted on the turret. In these experiments, the turret was kept at a constant orientation to provide a "compass sense". Exploration was carried out using the separate translational and rotational motors located in the base of the robot.

the rotational motor was controlled independently, and the turret left stationary, in order to provide a consistent "compass sense". In other words, the robot explores its environment with its sensors always facing in the same global direction rather than the direction of travel. Thus, the appearance of locations to the robot depends on the robot's position alone, not its orientation. In the experiments conducted so far, no problem has been found with angular drift of the turret against the rotational movement of the robot, though this will need to be investigated thoroughly in subsequent large scale testing.

The system was implemented as hierarchy of behaviours, as shown in figure 1. The implementation exploits the fact that FortyTwo's Linux operating system allows multi-tasking, enabling different behaviours to be programmed independently and executed as separate, concurrent processes. Appropriate parameter settings for the various components of the system were determined experimentally, as given in Appendix B. Attention is drawn to these "magic numbers" in the intepretation of results wherever their effect was found to be most critical.

Two different exploration behaviours, wall following and wandering (both incorporating obstacle avoidance), were implemented in a Perceptron-like neural network, which was trained using instinct rules (see Nehmzow [13] for details). At the moment, the controller is completely reactive, directly associating different perceptions from the sonar and infrared sensors with appropriate actions for the translational and rotational motors.

In each experiment or simulation, the localisation error is calculated as the difference between the actual position of the robot and the winning position estimate created by the localisation algorithm. Where two or more hypotheses share the same confidence level, the "worst case" error is taken from the hypothesis which lies furthest from the actual location.

6.2 Real Robot Experiments

The system was tested on FortyTwo in the Robotics Laboratory at Manchester University. The wall following behaviour was used for exploration, and the ART2 network configured to receive input from the robot's 16 infra-red sensors. Map building was carried out during a single circuit of an enclosure

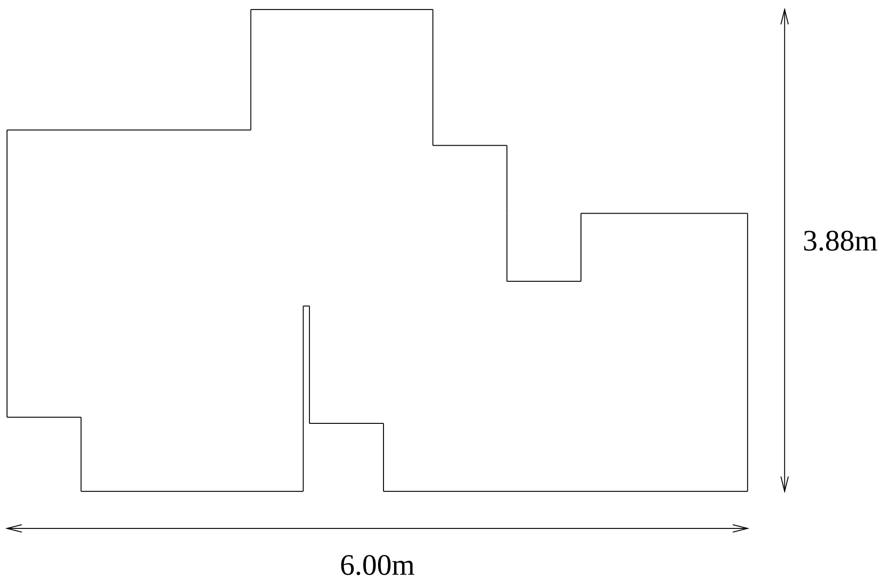

Figure 8: *World Used for Real Robot Tests.* A clockwise wall follower was used for exploration, and the robot's infrared sensors were used for input to the ART2 network.

made from walls and boxes, which is shown in figure 8. This was purposely designed to contain areas of perceptual aliasing, as shown in the map (figure 9). The average forward speed of the robot travelling around this enclosure was 0.10 ms^{-1}.

The algorithm was run for 20 iterations per trial for 30 trials. The results given in figure 10 show a steady downward trend in the mean localisation error over time. The "actual" position of the robot had to be taken from global odometry, because no external means of recording FortyTwo's real position is currently available. The results shown will therefore only be as accurate as the drift in odometry will allow.

In addition, the time and number of iterations through the algorithm taken for the localisation error to fall below the distance threshold D was measured in each trial. The mean number of steps was 7.0, with a standard deviation of 5.6. The mean time taken was 27.3 seconds, with a standard deviation of 21.9 seconds. (The average curve shown in figure 10 takes longer than this to reach the D level because this calculation takes into account the worst case localisation errors as explained previously.)

Inspection of individual localisation attempts showed that the algorithm was able to localise successfully once the robot had travelled through a unique sequence of perceptions along the perimeter of the enclosure. For example, following the route taken by the robot clockwise in figure 9, the sequence "0, 1, 5" of ART2 categories uniquely identifies the point at (-1.28, 2.81). However, the sequence "0, 1" would be ambiguous, because there are two possible places in the route where this occurs.

While it was possible to make such qualitative judgements about the behaviour of the localisation system on the real robot, the use of global odometry for calculating the localisation error makes it very difficult to make a really accurate assessment on the performance of the localisation algorithm. It is also very difficult to separate different possible sources of error on the real robot, e.g., sensor noise, wheel slippage, chaotic variation in the exploration strategy, etc. For this reason, and for this reason only, the following tests were conducted on the simulator, where different error levels could be carefully controlled.

6.3 Simulation - Wall Following

In each of these simulations, the localisation error was recorded against time over 30 trials, and mean error curves were plotted. In addition, the time and number of iterations through the algorithm taken

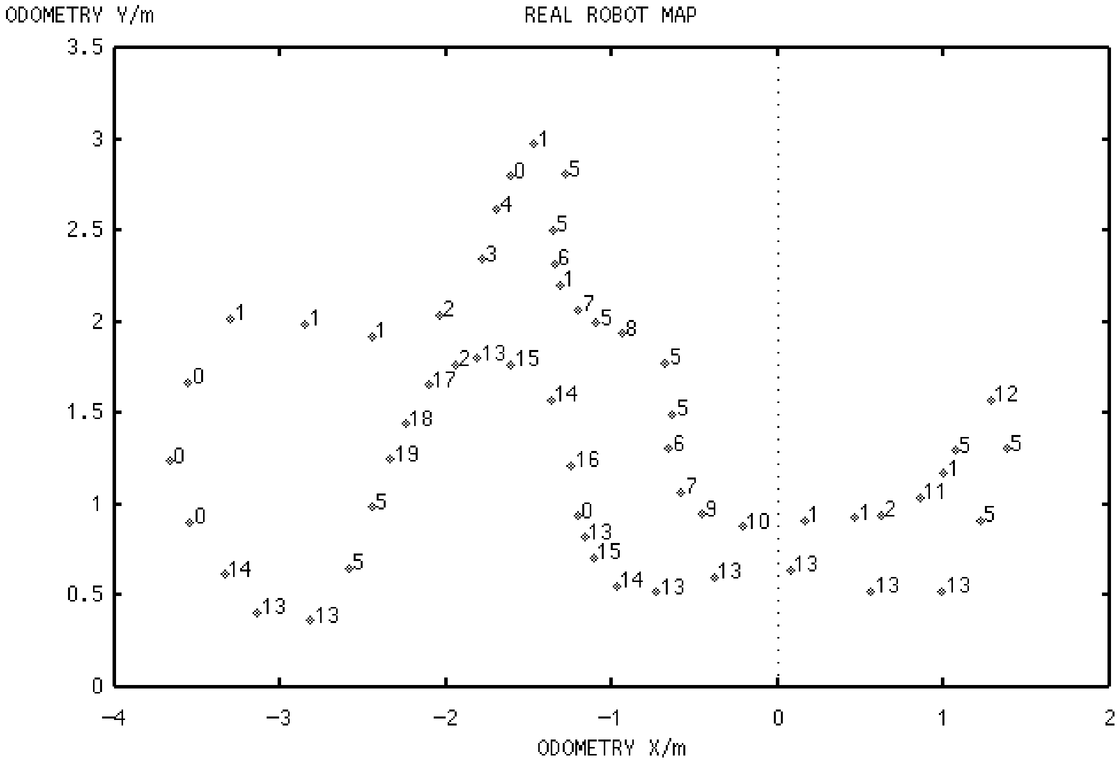

Figure 9: *Map Created By Real Robot.* The location points follow the path of the wall following robot around the enclosure shown in figure 8

.

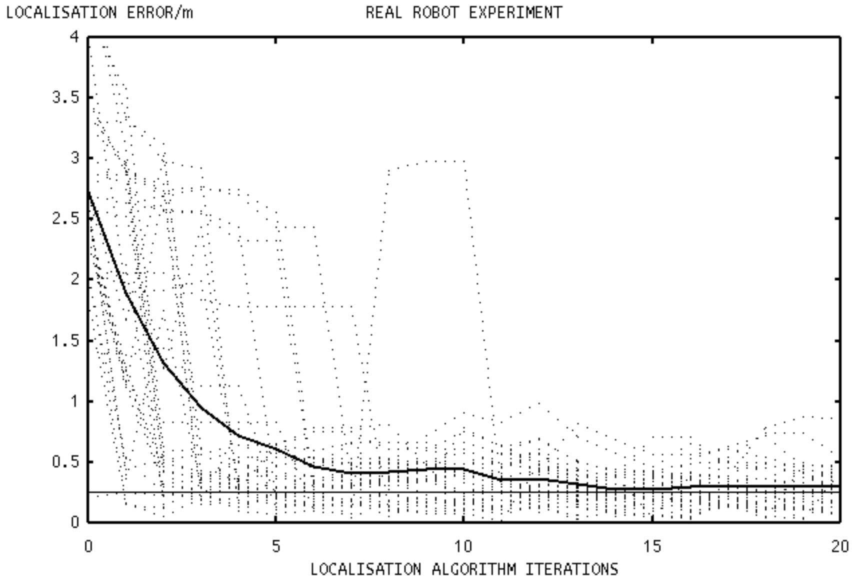

Figure 10: *Localisation Error on Real Robot.* The thick line shows the mean error taken over 30 trials. The dotted lines show the individual trials, and the horizontal line at $y = 0.25$ denotes the distance threshold D used during map building.

for the localisation error to fall below the distance threshold D was also measured for each trial. A statistical analysis of these results is presented in table 1.

The times used were taken from the UNIX system clock during the running of the simulation, and are considerably longer than those found on the real robot. The comparatively slow speed of execution was due to the fact that all four simulations described here were run concurrently, using a single simulated robot to provide the sensory input for each of the localisation programs. Thus, all results are based on the same sensor readings taken from the same simulated robot over the same time period, using the same exploration behaviour.

Simulated Error Conditions		**Steps**	**Taken**			**Time**	**Taken**	**/s**
	Mean	S.D.	Min	Max	Mean	S.D.	Min	Max
No Error	3.6	1.1	1	6	169.6	67.7	22.5	313.5
10% Classification Error	12.4	9.0	3	41	195.3	108.4	58.2	518.3
25% Drift Error	4.0	1.3	1	7	175.2	67.1	26.0	314.3
Combined Errors	11.4	6.5	2	31	184.8	121.6	39.9	692.4

Table 1: *Comparison of Localisation Times in the Wall Following Simulations.* Here the localisation time is recorded as the time taken for the localisation error to fall below the distance threshold D used in map building. Similarly, the number of iterations through the algorithm taken were recorded in each trial. The results are taken over the 30 trials carried out for each simulation.

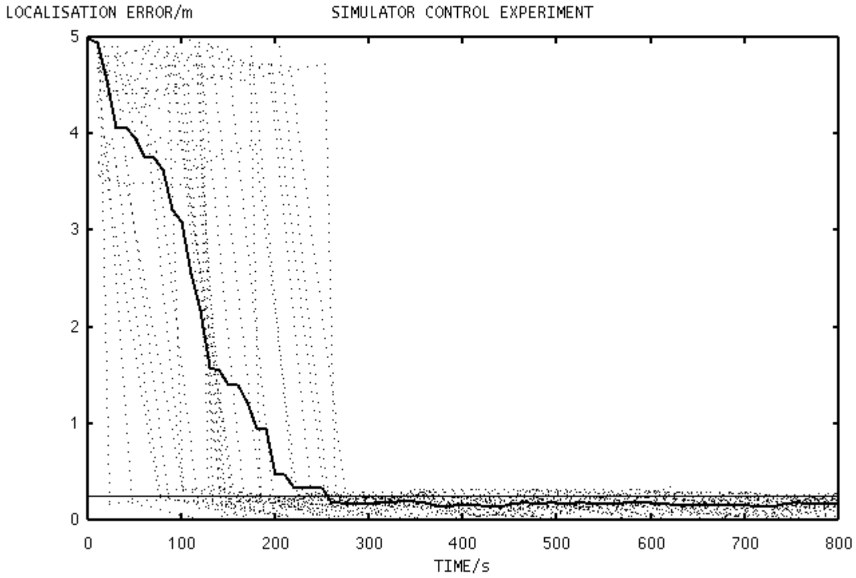

Figure 11: *Simulated Wall Follower with No Errors Added (Control).* The thick line shows the mean error taken over 30 trials. The dotted lines show the individual trials, and the horizontal line at $y = 0.25$ denotes the distance threshold D used during map building.

6.3.1 Control (No Errors Added)

A test environment was built on the simulator, as shown previously in figures 4 and 6. This was deliberately designed so that no single place had a unique perceptual signature, a "worst case scenario" for mobile robot localisation. Figure 11 shows the resulting localisation error, averaged over 30 trials. It can be seen that after about 250 seconds, the mean error curve falls below 100 cm, the distance threshold D used in the map building process. Thus, it can be seen that the algorithm will always localise to the nearest point in the map when no errors are present in the system.

Another important observation was made here. Referring to figure 6, imagine two wall following robots, one starting off around the origin, and the other on the far left of the map. Both would experience the same sequence of changes to the currently perceived ART category as they travelled clockwise around the enclosure, namely "0, 1, 2, 3, 0, 4, 5, 6, 0 ...". If the localisation algorithm really could only succeed after a unique sequence of ART categories had been covered, then it would only be at the next juncture, where the first robot would record a "4" and the second a "1", that their respective locations could be properly disambiguated. However, the simulation conducted here showed that this was not the case!

In fact, both simulated robots would find a winning location hypothesis emerging somewhere around "0, 1, 2, 3 ...", where the corresponding perceptual regions shown in figure 4 vary significantly in size. Thus, it is not only the sequence of ART categories perceived which determines the robot's estimated position, but also the *size* of the corresponding regions in the map. Thus, the robot continues to accumulate useful information even when no change is detected by the ART network. This aspect was never deliberately "programmed" into the system, rather it emerges from the use of relative odometry for comparing successive hypotheses in the localisation algorithm.

6.3.2 Classification Error

In this simulation, a 10% classification error was added to the ART network. Every time the current ART category was sampled, a one in ten chance of misclassification was introduced, where one of the other possible categories was randomly selected. Again, this is a worst case scenario. In practice, the

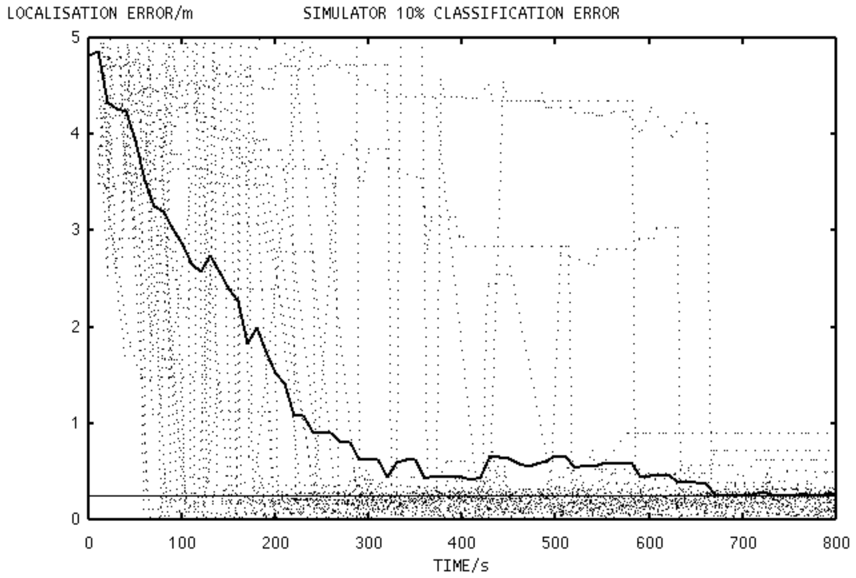

Figure 12: *Simulated Wall Follower with 10% Classification Error.* Mean error taken over 30 trials.

classification error rate is much lower than this on FortyTwo, although this is hard to quantify for reasons given previously.

Results are shown in figure 12. Although the downward trend is prevalent once again, the mean error curve takes around 650 seconds to reach the 100cm level. Again, however this is a worst case average, taking into account a steadily declining number of "rogue" values, where a bad hypothesis has made a temporary rise in confidence over the eventual winner. The rogue values are gradually suppressed as the best hypothesis picks up enough "momentum" due to the gain term used in the algorithm.

Clearly, the balance between the gain and decay terms used in the algorithm is important here; if the overall gain is too low then the good hypotheses may never emerge, but if it is too high then the rogue hypotheses may take a long time to be overtaken. A separate set of simulations showed that an appropriate level for the gain factor is around $1.5 \leq GAIN \leq 8$, with a decay factor of around $0.6 \leq DECAY \leq 0.8$, given that hypotheses are deleted when their confidence falls below a minimum level $MIN = 0.5$.

Table 1 shows that although the mean time taken to localise was only a bit higher than in the control, it took a lot more steps through the algorithm. This is because each time the localiser looks at the current ART category, there is a 1 in 10 chance that a misclassification will trigger another iteration (in fact a 2 in 10 chance, because the algorithm will be triggered again when the correct classification is obtained once more.)

6.3.3 Drift Error

Here, an artificial global drift in odometry was deliberately introduced. This was done by integrating the distance travelled by the robot, and then adding the required percentage of this to the absolute odometry readings in a drift direction randomly chosen at the start of each trial.

In a separate set of simulations, it was found that even with a drift error of up to 20%, the performance of the localisation algorithm was almost indistinguishable from the control. Introducing a 25% drift error had some impact on performance, as shown in figure 13. Here it can be seen that some of the winning hypothesis became corrupted by drift and took a while to decay before being replaced by another correct hypothesis. The distance threshold T used in the matching part of localisation algorithm becomes critical here; if it is too low then the matching process will fail, if it is too high then the algorithm will never

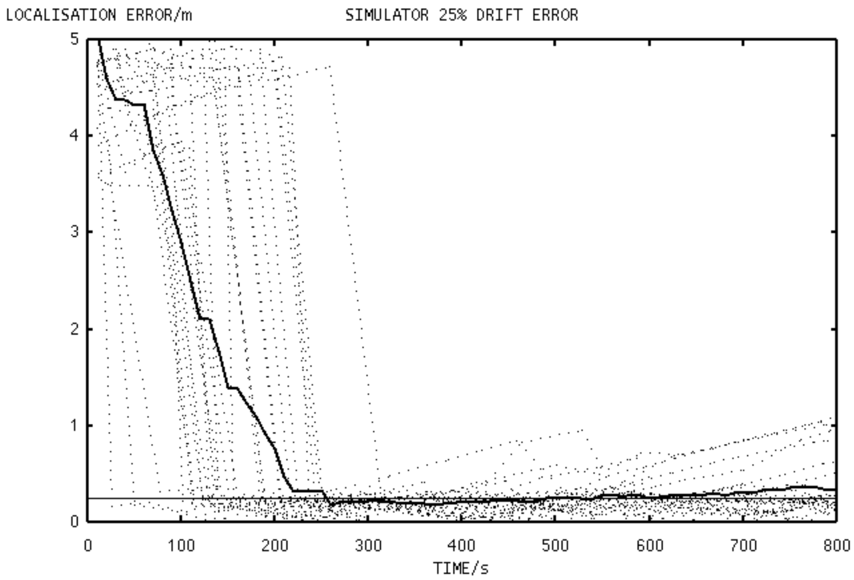

Figure 13: *Simulated Wall Follower with 25% Drift Error.* Mean error taken over 30 trials.

discriminate between good and bad hypotheses. A value of $2D$, twice the distance threshold used in map building, was found to be sufficient for normal operation.

However, it should be noted that the resilience to drift error is not particularly surprising given that these simulations have used a "perfect" map and that the algorithm only uses relative odometry. As far as the localiser is concerned, the metric information contained in the map is only used as a rough measure of proximity, so local distortions in the map should not be a problem. A big problem yet to be addressed is how to cope with drift error during map building, i.e., how to make sure that the map will be globally consistent. For example, with a drift in odometry, the same location could be represented twice because one of the criteria for adding locations to the map is a sufficiently distinct position in Cartesian space to other, already mapped locations. This criterion would obviously be met if locations are indeed distinct, but also if they appear to be distinct due to drift error. Possible solutions to the global drift problem during map building are discussed in the conclusions.

6.3.4 Combined Classification and Drift Error

A fourth simulation revealed a particularly interesting synergy arising from the combination of two different error effects. Both the 10% classification error and the 25% drift error were applied. Figure 14 shows both the rogue classification and drift effects discussed previously. However, what is unusual about these results is the relative comparison with the other simulations in table 1, particularly against the 10% classification error on its own. It can be seen that adding the drift error actually made the algorithm localise in fewer steps and less time!

This can be explained as follows; the 10% classification error results in rogue hypotheses gaining confidence as described before. Under certain circumstances, they will gain in confidence when the misclassifications (and classifications) of the ART network produce more overall evidence in their favour than against. Given that the evidence accumulation step of the algorithm favours hypotheses which stay within the bounds of the distance threshold T, the rogue hypotheses can therefore expect to be maintained longer if they can keep position during the steps where their confidence is being decreased. Therefore, accurate dead reckoning will actually allow the rogue hypotheses to survive longer than less accurate dead reckoning. Introducing the 25% drift error means that the rogue hypotheses decay faster than in the simulation with 10% classification error, in this case more than offsetting the negative impact

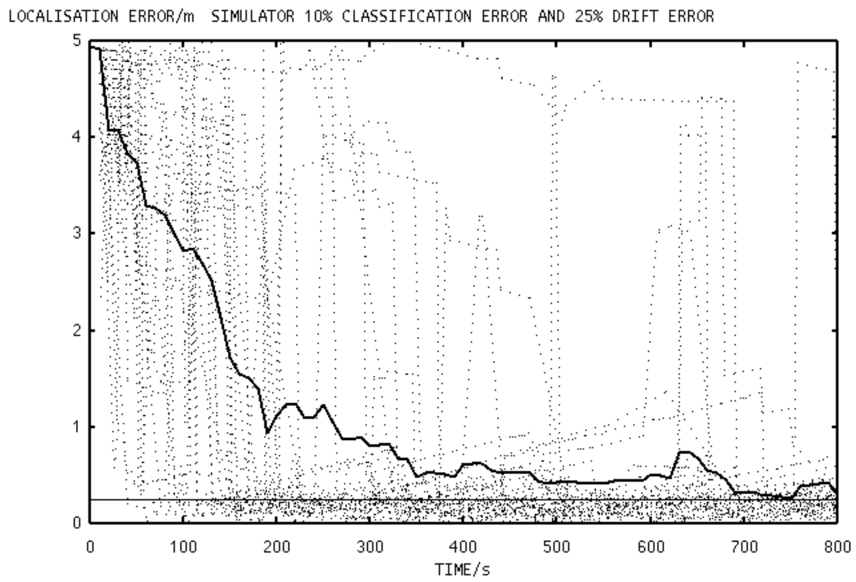

Figure 14: *Simulated Wall Follower with both 10% Classification Error and 25% Drift Error.* Mean error taken over 30 trials.

of the drift effect.

Apart from this interesting anomaly, figure 15 shows that there is little to separate the overall performance of the algorithm in the different simulations, showing a graceful degradation in performance with respect to error. The localisation algorithm is robust with respect to drift error, but this will need to be investigated on a bigger scale to evaluate the problem during map building. Classification errors do not seem to affect the performance against time too much, but the algorithm has to work harder to achieve this.

6.4 Simulation - Wandering ("Two Dimensional" Localisation)

In the previous sections, the wall follower was used for exploration. Route following is termed a one-dimensional exploration strategy here, because the robot can only move forwards or backwards along a fixed path. In contrast, a variable "two-dimensional" strategy was used here (see figure 16), where the same locations may be visited from many different directions. (Maintaining a steady turret orientation made recognition by ART2 possible here, as discussed before.)

Sonar was used for input to the ART2 network, as it was found that much of the space away from the walls of the enclosure contained no features discernible by infra-red. (An interesting feature of ART2 is that "flatline" patterns, i.e., those with no obvious features detectable above the accepted level of background noise, cannot be classified in any category.) Again, no single area of the environment was uniquely defined in perceptual space (see figure 17).

The wandering behaviour used was based on learned smooth, continuous obstacle avoidance (see [13] for details). To prevent the robot from getting stuck in loops around one part of the environment, random noise was added to the output vector of the neural controller. In addition, to cover different areas of the environment using this exploration strategy, and also to make the behaviour unpredictable, the sensitivity of the obstacle avoiding behaviour was changed at random intervals by adjusting a gain term applied to the rotational "motor neuron". Thus, this simulation made a good test of the robot's ability to localise without relying on visiting the same sequence of locations each time.

Even with the above extensions, the wandering robot could take a very long time to build a complete map in a small enclosure like the test environment. This is hardly surprising, as the exploration strategy

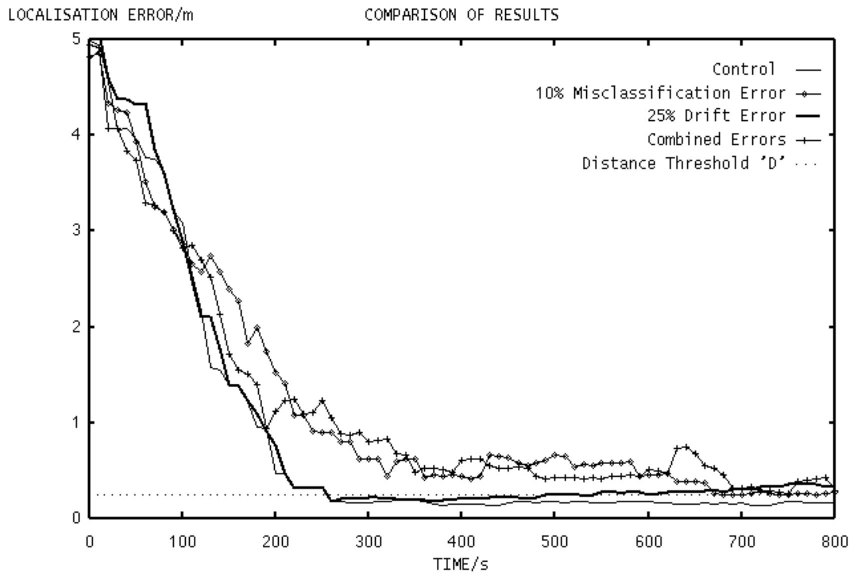

Figure 15: *Comparison of the Different Wall Following Simulations.* Mean errors taken over 30 trials each. An identical wall following behaviour was used throughout.

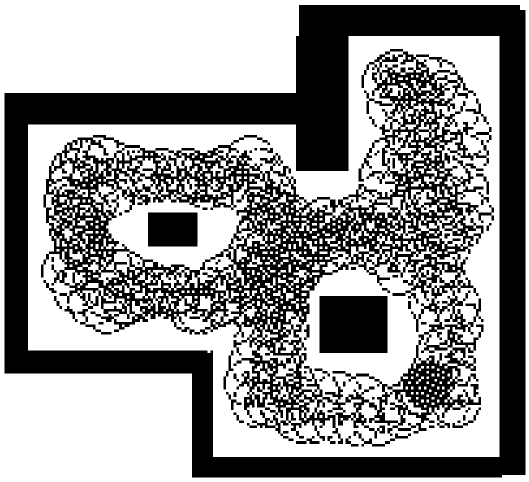

Figure 16: *Simulator Test Environment for "2-Dimensional" Localisation.* The trace taken from the simulator shows that the wandering robot may approach locations from arbitrary directions.

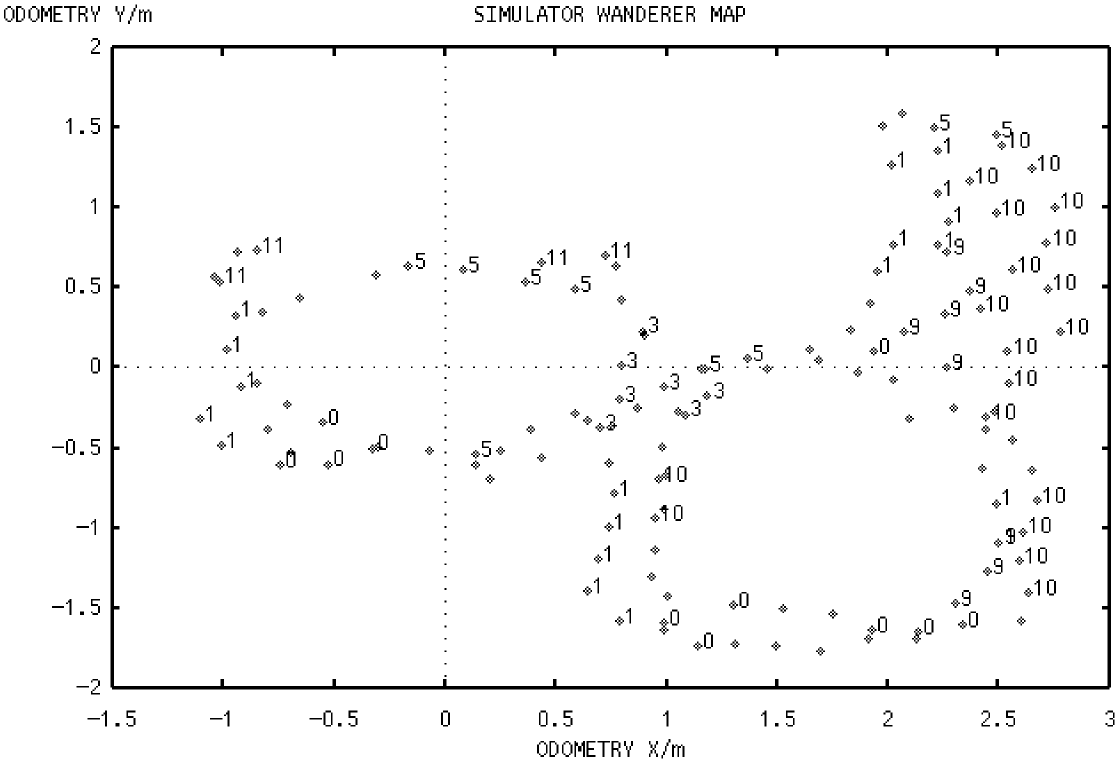

Figure 17: *Location Points Created by Simulated Wandering Robot.* Not all of the ART categories are shown here for clarity.

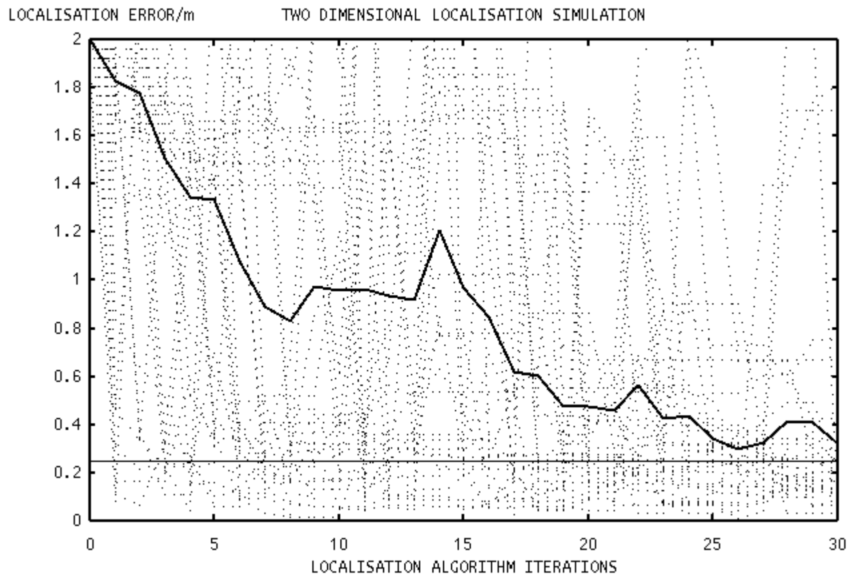

Figure 18: *Localisation Error for Wandering Simulation.* (Mean error taken over 30 trials).

chosen basically relies on blind chance and "the law of averages" to eventually steer the robot towards uncharted areas of the map.

This was also reflected in the results obtained (see figure 18). While the robot never failed to localise eventually, it could take a long time before it was able to find a perceptually unique path through the enclosure. The mean number of steps taken was 10.2, but there was a lot of variation around this level (standard deviation 7.5 steps), depending on the luck of the localiser in being party to a good piece of exploration by the wandering behaviour.

Despite the limitations discussed above, the wandering simulation does seem to verify the underlying principles of the localisation algorithm, providing a good existence proof of the ability to localise without following a rigid sequence of "landmarks". However, significantly better results were achieved in both map building and localisation when using the wall follower, indicating a big advantage in exploration using canonical paths over random movement (see also Nehmzow [14] for a more detailed discussion of this aspect). In particular, map building is far more efficient, using much fewer location points in a similar sized area. A map constructed from a single tour of the environment can contain all of the information needed by the lost robot to relocalise using the same exploration strategy. Localisation was also found to be much easier when following the same path as before.

7 Conclusions

7.1 Contribution to Mobile Robotics

A complete system has been presented for a mobile robot, which can localise even in environments where no single place has a unique perceptual signature. No prior estimate of position is required, so the robot can recover even after becoming completely lost. The localisation algorithm implements a competitive learning scheme, which is related to hippocampal models taken from biology. During an exploration phase, evidence is accumulated to support competing place estimates. Perceptual changes (from sonar, infrared and relative odometry) are correlated with an internal world model to provide positive or negative reinforcement of the emerging hypotheses.

The algorithm has some other interesting emergent properties. One is that the robot continues to collect useful evidence even when moving between location points which share the same perceptual signature. More generally, it can be seen that a winner emerges once a perceptually unique path has been traced through the map. This applies regardless of the length of the path, or in other words, the amount of past sensory experience required to disambiguate similar looking locations. In principle, the algorithm therefore generalises to cover arbitrary levels of perceptual aliasing in the environment, provided that the following conditions are upheld;

1. That the environment has a finite size (i.e., that the environment has boundaries which are perceptually distinctive from its interior).
2. That the robot has a compass sense, or can recover its orientation unambiguously from landmarks in the environment.
3. That the exploration strategy of the robot will eventually find a perceptually unique path through the environment. (Conditions 1 and 2 guarantee that such a path will exist.)

The reasoning behind this is as follows; if the environment has a finite size, then by heading in any direction, the robot will eventually find one of its boundaries. Following the boundary of the enclosure would eventually allow the robot to find its location, provided that it could distinguish repeating patterns of perceptions around the enclosure. In the worst case of a circular or radially symmetric room, the robot would need a compass to find its location. Otherwise, the relative position of landmarks should allow the robot to recover its original orientation.

Of course, this does assume a perfect world; in practice, the above assumptions may be undermined to some extent by the inherent uncertainties of using a real robot in the real world. However, the experiments and simulations conducted showed that the localisation algorithm presented here is very robust, only experiencing a graceful degradation in performance with respect to the errors introduced. With regard to the third condition, the results also demonstrated the superiority of following canonical paths over random exploration.

To summarise, the key achievements of the work are the handling of perceptual aliasing and the ability to localise when no predetermined estimate of position is available. The underlying principles are the maintenance of competing location hypotheses and the incremental accumulation of evidence over time.

7.2 Outstanding Problems

The problems found in the existing system are briefly restated below, together with some outstanding requirements which must also be addressed in future work. Some potential solutions are discussed in more detail in the following section.

- *Global Drift Error.* The map building process is currently dependent on global odometry for providing the metric information used in the map. Similarly, the issue of angular drift of the robot's turret may need to be addressed, because the system is currently reliant on accurate servoing to keep the turret facing in a constant direction.
- *Dynamic Environments.* The system cannot cope with changes to the environment, as the robot's map will no longer be correct.
- *Inefficient Exploration.* The exploration behaviours used are incapable of "purposeful" activity, e.g., directing the robot towards unexplored areas of the map instead of wandering aimlessly, as demonstrated in the wandering localisation simulation.
- *Localised Sensory Perceptions.* The recognition system used (ART2) can only recognise previously recorded perceptual signatures from more or less the same physical position. Thus, the robot must move across nearly all of the Cartesian space in an environment to build a useful map. This can be very time–consuming, as demonstrated in the wandering simulation. It would be much better if the robot only needed to scan a room or corridor from a few different positions to adequately map the entire space.

7.3 Directions for Future Research

7.3.1 Lifelong Learning

Implementing lifelong learning, whereby both map building and localisation would both be run continuously and in parallel, might provide solutions to two of the above problems. Running the map building process in parallel with the localiser would mean that the system would be able to cope with lasting changes to the environment. Conversely, running the localisation process in parallel with the map-builder would help to eliminate global odometry slippage during map building.

In terms of the overall system architecture, this would mean putting a link back from the localiser to the dead-reckoner (see figure 1). In addition, the ART prototypes would have to be fixed after being created to prevent the problem of overtraining, as described in section 3.4.

The map building process would need to more flexible, incorporating deletion as well as addition to the map, perhaps utilising variable confidence levels as in Zimmer [23] or Yamauchi [22]. However, distinguishing transient changes in the environment, e.g., people walking past, people in wheelchairs, other robots, etc., would be a problem, as it would be undesirable to include such things in the map. Comparison of predicted and actual sensor patterns might provide a possible solution here.

7.3.2 Pro–Active Exploration

At the moment, the exploration behaviours used are purely reactive, and take no account of the navigation requirements of higher level behaviours. For example, as shown in the wandering simulation, it would be much better if the exploration could be directed towards "uncharted" areas of the map. Similarly, while it has been shown that localisation is possible after a perceptually unique path has been taken through the map, there is nothing in the exploration strategies used here to guarantee that this will be an optimal or shortest possible such path. It would be much better if the robot could purposely explore in order to disambiguate the current location signature as quickly as possible.

This transition from aimless to purposeful behaviour might be termed "pro–active" as opposed to "active" or "reactive" exploration. This might be achieved by realising some form of meta–level learning, perhaps learning to mediate between a set of previously acquired exploration behaviours. In terms of the overall system architecture, this would mean creating dependency links from the map–builder and the localiser to the explorer.

7.3.3 Variant Recognition

The current recognition system means that the robot must cover most of the space in its environment to build a useful map, as discussed above. At the moment, the classification of places using ART2 might be termed "invariant", because places can only be recognised by direct comparison of current and stored sensor patterns. It would be much better if some kind of transformation could be applied to the sensor patterns to allow enclosed spaces to be recognised from different view points. The robot might then only need to take a couple of "snapshots" in a given room to successfully map the whole of the surrounding space. Weiss and von Puttkamer [21] use cross-correlation of range-finder scans for this purpose. We propose to investigate this technique for the application discussed here.

7.3.4 Scaling It All Up

Along with the other related work reviewed earlier, the system developed has only been used in relatively small, carefully controlled test environments. While the localisation methodology presented has been shown to work "in principle", it is almost certain that new problems will be found in trying to scale this up to work on a larger scale. It will probably also be necessary to consider evidence from many different sources when the simple "blocks world" uniformity of the laboratory is left behind. Future work will therefore look at combining different sensor modalities and sensing strategies, e.g., using active vision as well as sonar, infra-red and relative odometry. The eventual aim of this research is to produce a robot which can localise effectively in a large, unstructured office environment, specifically the Computer Science building at the University of Manchester.

References

[1] S. Atiya and G. D. Hager, "Real–Time Vision–Based Robot Localization", IEEE Trans. Robotics and Automation, Vol. 9, No. 6, pp. 785-800, 1993.

[2] A. Bühlmeier, H. Dürer, J. Monnerjahn, M. Nölte and U. Nehmzow, "Learning by Tuition, Experiments with the Manchester 'FortyTwo' ", Dept. Computer Science, University of Manchester, Technical Report Series UMCS-96-1-2, 1996.

[3] P. Calter, "Technical mathematics with calculus", 3rd edition, Prentice-Hall, 1994.

[4] G. Carpenter and S. Grossberg, "ART2: Self–Organization of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics, Vol. 26, No. 23, pp. 4919-4930, 1987.

[5] T. Edlinger and G. Weiss, "Exploration, Navigation and Self–Localization in an Autonomous Mobile Robot", Autonome mobile Systeme '95, Karlsruhe, Germany, Nov 30 - Dec 1 1995.

[6] S. Engelson, "Passive map learning and visual place recognition", Ph.D. thesis, Dept. Computer Science, Yale University, New Haven, CT, 1994.

[7] S. Grossberg, "Neural Networks and Natural Intelligence", MIT Bradford Press, 1988.

[8] J. Horn and G. Schmidt, "Continuous localization of a mobile robot based on 3D laser range data, predicted sensor images, and dead–reckoning", Robotics and Autonomous Systems 14, pp. 99-118, 1995.

[9] B. J. Kuipers and Y.-T. Byun. "A Robust, Qualitative Method for Robot Spatial Learning", Proc. AAAI 1988.

[10] A. Kurz, "Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data", IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 26, No. 2, April 1996.

[11] D. C. Lee, "The Map–Building and Exploration Strategies of a Simple, Sonar- Equipped Mobile Robot; An Experimental, Qualitative Evaluation", Ph.D. thesis, University College London, England, 1995.

[12] M. J. Mataric, "Integration of Representation Into Goal–Driven Behaviour–Based Robots", IEEE Trans. Robotics and Automation, Vol. 8, No. 3, June 1992.

[13] U. Nehmzow, "Experiments in Competence Acquisition for Autonomous Mobile Robots", PhD thesis, Dept. Artificial Intelligence, University of Edinburgh, 1992.

[14] U. Nehmzow, "Animal and Robot Navigation", Robotics and Autonomous Systems, Vol. 15, No. 1 - 2, pp. 71-81, 1995.

[15] U. Nehmzow and T. Smithers, "Mapbuilding using Self–Organising Networks", in J. A. Meyer and S. Wilson (eds.), "From Animals to Animats", Proc. SAB90 Paris, pp. 152-159, MIT Press, 1991.

[16] U. Nehmzow, T. Smithers and J. Hallam, "Location Recognition in a Mobile Robot Using Self-Organising Feature Maps", in G. Schmidt (ed.), "Information Processing in Autonomous Mobile Robots", Springer Verlag, 1991.

[17] U. Nehmzow and T. Mitchell, "The Prospective Student's Introduction to the Robot Learning Problem", Dept. Computer Science, University of Manchester, Technical Report Series UMCS-95-12-6, 1995.

[18] C. Owen, "Landmarks, Topological Maps and Robot Navigation", M.Sc. thesis, Dept. Computer Science, University of Manchester, England, 1996.

[19] M. Recce and K. D. Harris, "Memory for places: A navigational model in support of Marr's theory of hippocampal function", Dept. Anatomy and Developmental Biology, University College London, England. In press - Hippocampus, 1996.

[20] J. Tani and N. Fukumara, "Learning Goal–Directed Sensory–Based Navigation of a Mobile Robot", Neural Networks, Vol. 7, No. 3, pp. 553-563, 1994.

[21] G. Weiss and E. v. Puttkamer, "A Map Based on Laserscans Without Geometric Interpretation", Intelligent Autonomous Systems-4 (IAS-4), Karlsruhe, Germany, March 1995, pp. 403-407.

[22] B. Yamauchi and R. Beer, "Spatial Learning for Navigation in Dynamic Environments", IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 26, No. 3, June 1996.

[23] U. R. Zimmer, "Self–Localization in Dynamic Environments", IEEE/SOFT International Workshop BIES'95, Tokyo, Japan, May 30-31 1995.

A Details of ART2 Architecture and Equations

This appendix gives all of the architectural details and equations required to build the ART2 network used in this paper (see figure 19). In ART2, the F1 units are implemented as six individual nodes, among which much of the control processing is distributed. A pre-processing layer F0 was added to further enhance the contrast and reduce noise in the input patterns. The mechanism used for updating the weight vectors was based on the implementation of ART2 by Paolo Gaudiano at the CMU Artificial Intelligence Repository [4].

A.1 F0 Layer Equations

The equations describing the dynamics of the F0 and F1 levels are given as follows, in order of bottom-up flow through the network. I is the input vector presented to the network, a, b, c, d, e and θ are constants (see section A.6), and $f(x)$ is a noise filtering function (see section A.3).

$$w'_i = I_i + au'_i$$

$$x'_i = \frac{w'_i}{e + \|w'\|}$$

$$v'_i = f(x_i)$$

$$u'_i = \frac{v'_i}{e + \|v'\|}$$

A.2 F1 Layer Equations

Similarly, the equations describing the dynamics of the F1 layer are given as follows, again in order of bottom-up flow through the network.

$$w_i = u'_i + au_i$$

$$x_i = \frac{w_i}{e + \|w\|}$$

$$v_i = f(x_i) + bf(q_i)$$

$$u_i = \frac{v_i}{e + \|v\|}$$

$$p_i = \begin{cases} u_i & \text{if F2 is inactive} \\ u_i + dz_{Ji} & \text{if the } J^{th} \text{ F2 node is active} \end{cases}$$

$$q_i = \frac{p_i}{e + \|p\|}$$

[4] `http://www.cs.cmu.edu/Groups/AI/html/repository.html`

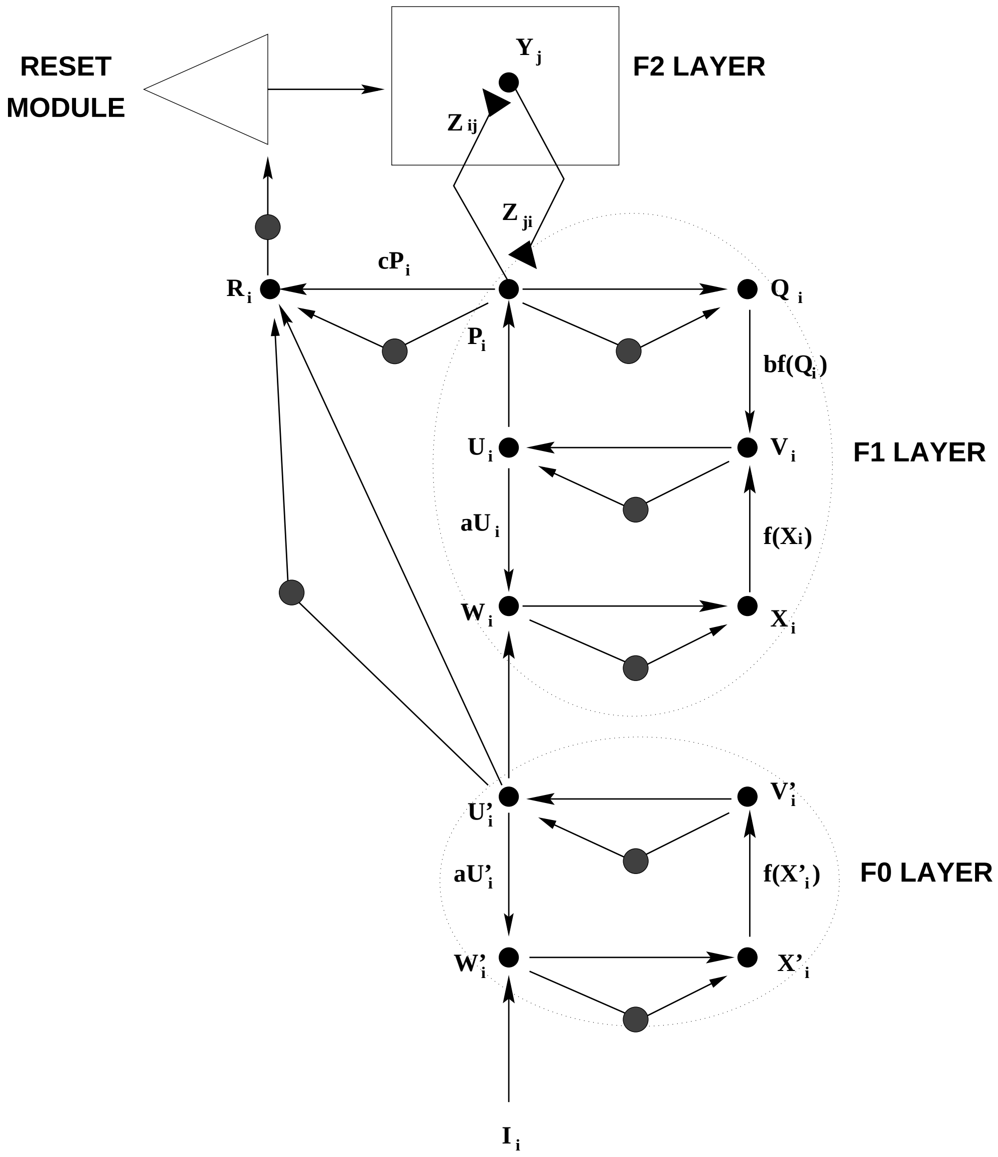

Figure 19: *ART2 Architecture* This shows the F2 layer (node Y), F1 layer (nodes P, Q, U, V, W and X), F0 layer (nodes U', V', W' and X'), reset module, feedforward weights (Zij) and feedback weights (Zji), for a single element, i, in the input vector, I. The arrows show processing steps between the nodes, and those with a shaded circle denote a normalisation operation on the associated vector. Adapted from Carpenter and Grossberg [4], figure 10.

A.3 Noise Filter

The following equation was used to cut out background "noise" below a given threshold θ at nodes v_i' and v_i;

$$f(x) = \begin{cases} 0 & \text{if } 0 \le x < \theta \\ x & \text{if } x > \theta \end{cases}$$

A.4 Weight Vectors

The bottom-up (z_{ij}) and top-down (z_{ji}) weights between layers F1 and F2 are initialised as follows, where M is the number of units in the F1 layer.

$$z_{ij} = \frac{1}{(1-d) \times \sqrt{M}} \quad \text{for all units } i \text{ in F1, } j \text{ in F2}$$

$$z_{ji} = 0 \quad \text{for all units } j \text{ in F2, } i \text{ in F1}$$

The following differential equations were used to adjust the weights for the winning F2 unit J, and these were solved using the Runge-Kutta method [3].

$$\frac{dz_{iJ}}{dt} = d(1-d)\left[\frac{u_i}{1-d} - z_{iJ}\right]$$

$$\frac{dz_{Ji}}{dt} = d(1-d)\left[\frac{u_i}{1-d} - z_{Ji}\right]$$

A.5 Reset Module

The activation of the reset module r is determined using the following equation;

$$r_i = \frac{u_i' + cp_i}{e + \|u'\| + \|cp\|}$$

The F2 layer is reset when the following criteria is met after the feedback phase, given the vigilance parameter ρ;

$$\frac{\rho}{e + \|r\|} > 1$$

A.6 Parameter Values

The following parameter values were used throughout all of the experiments and simulations documented in this paper;

$$a = 5.0, b = 5.0, c = 0.225, d = 0.8, e = 0.0001, \theta = 0.3$$

Different values were used for the vigilance threshold ρ depending on the particular experiment or simulation being carried out (see also next appendix). On both the real robot experiments and the simulations using infrared input, $\rho = 0.9$ was used. In the simulations using sonar, $\rho = 0.75$ was used. In general, the value of the vigilance parameter was found to have a critical effect within the range of approximately $0.7 \le \rho < 1.0$, where high values result in a fine categorisation and low values in a coarse categorisation.

B Parameters (Magic Numbers)

The following table shows the values of the variable parameters or "magic numbers" used in the various components of the localisation system during the different experiments and simulations.

Parameter	**Description**	Value in Real Robot Experiments	Value in Wall Follower Simulations	Value in 2-Dimensional Simulation
ρ	Vigilance threshold used in ART2 network	0.9	0.9	0.75
D	Distance threshold used in map building	0.25m	0.25m	0.25m
$GAIN$	Gain factor used in localisation	8.0	3.0	3.0
$DECAY$	Decay factor used in localisation	0.7	0.7	0.7
MIN	Minimum confidence level allowed in localisation	0.5	0.5	0.5
T	Match distance used in localisation	0.50m	0.50m	0.50m