Feature-based Mapping in Real, Large Scale Environments using an Ultrasonic Array[1]

Kok Seng CHONG

Institute of Microelectronics,

11 Science Park Road,

Singapore Science Park II,

Singapore 117685

Lindsay KLEEMAN

Intelligent Robotics Research Centre

Dept of Elect and Computer Systems Eng.

Monash University, Clayton Victoria 3168,

Australia

Abstract

This paper presents a strategy for achieving practical mapping navigation using a wheeled mobile robot equipped with an advanced sonar sensor. The original mapping navigation experiment, carried out with the same robot configuration, builds a feature map consisting of commonplace indoor landmarks crucial for localisation, namely planes, corners and edges. The map exhaustively maintains covariance matrices among all features, thus presents a time and memory impediment to practical navigation in large environments. The new local mapping strategy proposed here breaks down a large environment into a topology of local regions, only maintaining the covariance among features in the same local region, and the covariance among local maps. This notion of two hierarchy representation drastically improves the memory and processing time requirements of the original global approach, while preserving the statistical details, in the authors' opinions, necessary for an accurate map and prolonged navigation. The new local mapping scheme also extends the endeavour towards reducing error accumulation made in the global mapping strategy by eliminating errors accumulated between visits to the same part of the environment. This is achieved with a map matching strategy developed exclusively for the advanced sonar sensor employed. The local mapping strategy has been tested in two large, real life indoor environments and successful results are reported here.

[1] This work was performed whilst K S Chong was at Monash University as a PhD student.

1 Introduction

This work is the fifth stage of a sonar robot project. Briefly, the first stage covers the design, calibration and error modelling of a highly accurate odometry design which features a pair of knife-edged, unloaded odometry wheels (Chong and Kleeman 1996i, Chong and Kleeman 1997i). In stage two, a sonar robot employing this odometry system is deployed to perform mapping in a structured, indoor environment (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii). The robot makes use of an advanced sonar sensor (Kleeman and Kuc 1995) capable of localising and classifying simple indoor features into planes, corners and edges. The introduced mapping strategy fuses planes, corners and edges together to generate a realistic feature map, using the Iterated Extended Kalman Filter (IEKF) and Julier-Uhlmann-Durrant-Whyte Kalman Filter (JUDKF), by exploiting the correspondence constraints (plane to plane, corner to corner or edge to edge) and relational constraints (corner to two planes) among features. A method for identifying phantom targets is also described. This is followed by stage three which implements a dual representation strategy to enable the robot to systematically explore an imperfectly structured environment while simultaneously running map building (Chong and Kleeman 1996iii, Chong and Kleeman 1997iii). The proposed exploration scheme exploits the strengths of grid maps and feature maps to overcome perceptive limitations of sonar to achieve obstacle avoidance and complete floorplan coverage. The last stage of this project is a map matching algorithm which enables a robot to localise itself from a previously generated map (Chong and Kleeman 1996iv), so that if the robot navigates in a previously mapped environment, map building can be augmented with new measurements. The algorithm is robust against

imperfect maps and phantom targets, and generates the minimum least square error estimate from all matched features.

The motivation behind this work is the need to extend the mapping strategy to large environments. The original strategy works well in a reasonably sized indoor environment where the number of landmarks to be mapped is limited. However, to map a large environment, such as a building with rooms and corridors, the strategy is challenged by the need to store and process an ever expanding list of features. This also implies gradual degradation of processing **speed** and increased demand for computer random access **memory** (RAM). For example, when a map is large and a new measurement arrives, there are many features that need to be checked for the possibility of fusion before the measurement can be considered a new feature. If fusion is possible, the robot position and the state of all, n say, existing map features must be updated with as many as $(n+1)(n+2)/2$ covariance matrices updated. Even if fusion does not occur, the new feature has an associated correlation matrix with every existing map feature. The requirement for memory grows quadratically and without bound. In (Leonard and Durrant-Whyte 1991) the size of covariance matrix is reduced by declaring features with a small covariance as 'confirmed targets', thereby deleting their cross-covariance with the robot position and other features.

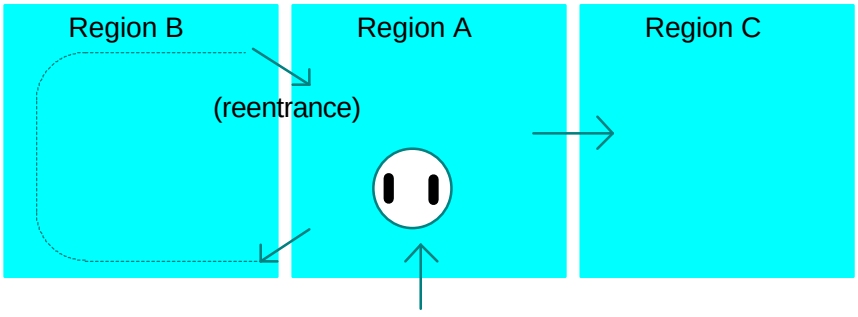

Figure 1: Robot exploring a large environment consisting of several 'regions'

Apart from the issues of computational speed and memory, the original mapping strategy cannot deal with the problem of **error accumulation** with complete success. There are several reasons why the global strategy does not work optimally. It is well known that linear approximations of equations can introduce bias (Jazwinski 1970, Leonard and Durrant-Whyte 1991, Bar-Shalom and Li 1993). Bias is also due to the modelling of the physical system, such as imperfect calibration of wheels, the assumed positioning of the sonar sensor at the centre of the robot, and the echo sampling electronics. In addition, error statistics of odometry position estimation, speed of sound variations and ultrasonic bearing measurement cannot be characterised accurately in all circumstances. All of these factors make continual consistent map generation difficult. Consider the example in Figure 1. The robot has just mapped region A, and its path planning leads it to explore region B first and then region C by traversing region A again. Theoretically speaking, the robot would localise with the features in region A to recover from the error accumulated in region B. In practice, if region B is large, the error accumulated may be large, so the robot may not be able to recognise that it is re-entering region A if it relies solely on an IEKF (whose linearisation assumes small state errors) and a χ^2 test. The robot may then proceed to build a conflicting map on top of the original map of region A. By the time region C has been mapped, the actual spatial positioning relative to region A is grossly inaccurate. This example illustrates a classical loop problem - if a robot returns to its starting position after prolonged navigation, can the starting position be identified? Loop problems are addressed in the context of Bayesian networks in (Pearl 1988, Chapter 4).

Another problem encountered in (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii) is that the IEKF, which improves state estimation based on several

'local iterations' (Jazwinski 1970), tends to **diverge** when the state covariance becomes too large. If an environment is large, the robot position covariance can easily exceed the working limit of the IEKF. Suppressing the growth of the robot position covariance is therefore important.

This work tackles the above problems using a local mapping strategy in which the environment is represented as a set of interconnected local maps. A new local map is created as usual until, after travelling a long distance, the robot covariance of position becomes large. The large position covariance is a good indication that subsequent features are not highly correlated with the earlier features in the local map. A new local map is begun. By not maintaining the covariance matrices between features from different local maps, memory and processing advantages are obtained. We also attempt to alleviate the inevitable problem of error accumulation by map matching (Chong and Kleeman 1996iv). Refer to Figure 1 again, the local mapping scheme attempts to recognise region A when the robot re-enters it by scrutinising the error in the robot position relative to region A. After affirming that the possibility of re-entrance is high, it performs a map match with new measurements on the local map of region A. If map matching is successful, the latest position of the robot will be accurate with respect to the local map of region A. The local map of region C will then be free of position errors accumulated during mapping of region B.

Local mapping is not a new concept. The work in (Asada, Fukui and Tsuji 1990), resorts to a graph of local feature maps that register the three dimensional information of the edges on the floor obtained from a camera. A new local map is created when none of the features in the current local map are visible from the next position. The arcs between local maps contain the robot displacement information. To facilitate fast local

mapping with sonar sensing, (Firby, Christianson and McDougal 1993) stores a local map as a list of pre-processed sonar reflection points. Dynamic environments are effortlessly accommodated by deleting conflicting points of reflections. However, none of the above approaches exploit re-observation of features to enhance the accuracy of the maintained maps, and are preliminary in the sense that they have not been tested in the difficult task of exploring large environments. Local mapping is not confined to feature based environment models. The work proposed in (Cassandra, Kaebling, and Kurien 1996), maintains a graph of local Bayesian distribution grid maps in its multi-sensory navigation scheme. Another version of the above approach is LOGnets (Malkin and Addanki 1990), which adopts a local occupancy grid with a special type of concentric, unevenly sized cells. Map update is a simple scheme of incrementing a cell frequency count when a sonar range combined with sonar pointing direction falls within a cell. In both grid based approaches, the methods for determining if a local map has been revisited are heuristic in nature.

One can regard the extreme opposite of the global approach as registering the measurements at every scan stop as a new local map. The problem with this idea is that one does not exploit the overlap between adjacent maps of features to reduce the uncertainties in the position of robot and features at all.

The useful properties of this local mapping system are:

- With the advanced sonar sensor (Kleeman and Kuc 1995), the robot can quickly extract the most salient natural landmarks useful for localisation.
- At any instant of navigation, the map, hence the processing demand, is small. This may allow future mapping based navigation to be performed as quickly as the robot can move.

- The generation of local maps is done with careful mathematical criteria, so the speed advantage is not gained at the expense of map integrity. As far as possible, re-observation of the same feature is exploited in map building as in the original global approach (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii).
- Through the process of map matching developed in (Chong and Kleeman 1996iv), the covariance of the robot position and its correlation with all the features in a previous local map can be correctly recovered. This facilitates further map building after re-entrance.
- In this paper the classical mapping problem of returning to the start after prolonged navigation is addressed in two challenging large environments, where the collaborative strength of map building and map matching are tested.

The paper is structured as follows. The processor, odometry system and sonar array of the robot are described in section 2. Section 3 summarises the global mapping approach that lays the groundwork for this work. A brief account of the map matching algorithm is presented in section 4. Section 5 presents the mathematical details of the proposed local mapping strategy. The experimental results are discussed in section 6. Limitations of this work are identified in section 7, followed by conclusions and future work.

2 The Architecture of the Mapping Robot

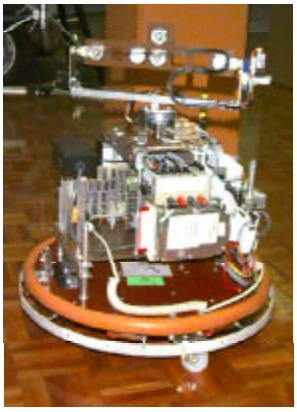

Figure 2: Werrimbi, the mapping robot.

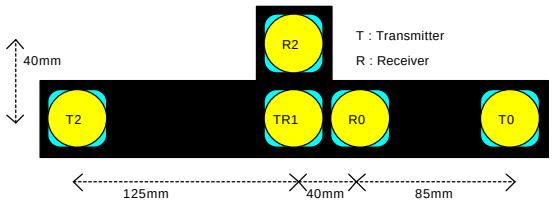

Figure 3: The sonar array configuration.

Detailed description of various hardware modules of the mapping robot, called Werrimbi (Figure 2), can be found in (Chong and Kleeman 1996i). Briefly, Werrimbi performs sensing with a highly accurate sonar sensing array (Figure 3). Ultrasonic pulses are transmitted from three transmitters in sequence and the received pulse samples on the three receivers are processed using a template matching scheme (Kleeman and Kuc 1995) to extract accurate time of flight information. The set of nine time of flight values are combined to obtain the reflector type (plane, corner or edge), range, horizontal bearing and vertical bearing. To perform scanning, the sensor is mounted on a pan mechanism. It first scouts rapidly for 360° to locate the bearings of potential targets using one transmitter and then returns to these directions to fire all three transmitters. The odometry system, as shown in Figure 4, consists of a pair of drive wheels and a pair of knife-edged encoder wheels mounted on vertical linear bearings. The knife-edged design reduces wheelbase (B) uncertainty while the linear

bearings remove loading on the wheels from the robot. Both the sensor and odometry are interfaced to a 486DX2 Intel processor card via an ISA Bus as shown in Figure 5.

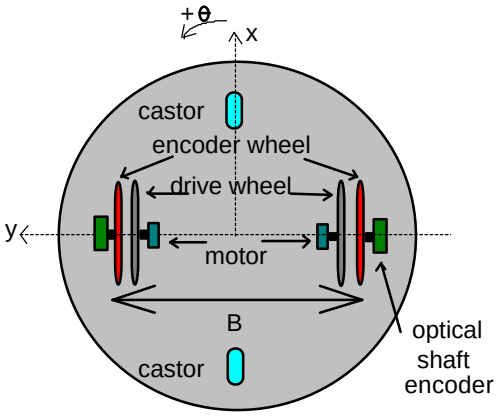

Figure 4 : The odometry system.

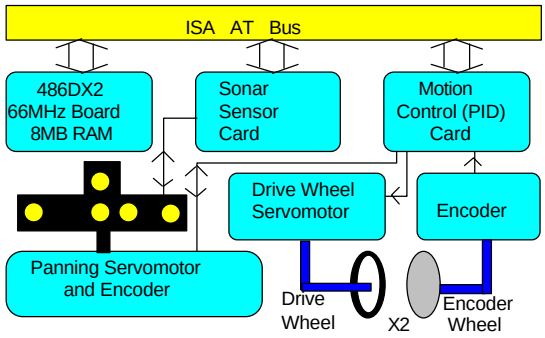

Figure 5 : The robot system architecture.

3 The Original Global Mapping Algorithm

The original global mapping algorithm can be found in (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii). To summarise, the environment is represented by a list of map primitives consisting of partial planes, corners and edges. The line parameters of partial planes, the Cartesian coordinates of corners and edges, the Cartesian coordinates of the robot position, the robot orientation and the speed of sound are collectively called the *state* of the map, among which *covariance* matrix 'blocks' are exhaustively maintained. When a new measurement is made, a search is launched for a map feature to perform fusion by generating a residual vector based on a collinearity constraint (plane to plane, corner to corner, edge to edge and one corner to two planes) between the new measurement and the fusion candidate. If the collinearity constraint passes a χ^2 test, an Iterated Extended Kalman Filter (IEKF) (Jazwinski 1970)

or a Julier-Uhlmann-Durrant-Whyte Kalman Filter (JUDKF) (Julier, Uhlmann and Durrant-Whyte 1995), are invoked which update all state parameters and covariance matrices based on the minimum mean square error. Both filters are in their own ways more superior than the more common Extended Kalman Filter (Bar-Shalom and Li 1993). In addition, the idea of Relocation-Fusion[2] advocated in (Moutarlier and Chatila 1989) is incorporated into both filters to reduce the impact of odometry bias accumulation. In the process of fusion, the endpoints of the existing partial planes in (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii) are constantly replaced by new endpoints to extend the partial planes to accurately depict long walls commonly observable in an indoor environment. If all χ^2 tests fail, the feature is integrated into the map. This infers that the list of state parameters is expanded and the covariance between the new feature and *all* existing map features are produced and stored. Periodically, the collinearity constraints among existing map features are tested to find the possibility of internal fusion, once again using χ^2 test. (Chong and Kleeman 1996ii, Chong and Kleeman 1997ii) also shows that the maps produced with IEKF and JUDKF are not markedly different. Even though JUDKF saves us from evaluating Jacobian matrices, the processing load is significantly higher than the IEKF.

As mentioned earlier, the two major factors affecting the speed of the local mapping algorithm are the computer memory requirements (RAM) and the processing demands. The following two subsections examine the two issues, and later on, use real experimental figures to compare the actual memory and processing demands of the

[2] The principle of Relocation Fusion is that all measurements are first used to update the $\mathbf{x}_0$ state (comprising the robot Cartesian coordinates, orientation and the speed of sound). The improved $\mathbf{x}_0$ is then used to re-compute the residual vectors and all related cross-correlations in order to update the map features.

global mapping approach and the local mapping approach, thereby justifying the latter approach.

3.1 Memory requirements.

This section investigates the memory requirements for storing a set of measurements and a map. The number of bytes required to store the attributes of all objects is tabulated in Table 1.

Table 1 : A list of objects used in mapping algorithm and their memory requirements.

Object stored	Quantity	Attributes/Purpose	RAM required by each object
partial plane, $\mathbf{x}_i$	n_p	line parameters	2×*sizeof*(double)
		two endpoints	4×*sizeof*(double)
		two termination status	2×*sizeof*(integer)
		object type	1
corner map feature, $\mathbf{x}_i$	n_c	Cartesian coordinates	2×*sizeof*(double)
		object type	1
edge map feature, $\mathbf{x}_i$	n_e	Cartesian coordinates	2×*sizeof*(double)
		object type	1
plane measurement	m_p	index to a partial plane	1×*sizeof*(integer)
		index to a robot position	1×*sizeof*(integer)
corner measurement	m_c	index to a state or two partial planes	2×*sizeof*(integer)
		index to a robot position	1×*sizeof*(integer)
edge measurement	m_e	index to a state	1×*sizeof*(integer)
		index to a robot position	1×*sizeof*(integer)
position history	n_{po}	Cartesian coordinates and orientation	3× *sizeof*(double)
current position,	1	Cartesian coordinates,	4×*sizeof*(double)

$\mathbf{x}_0$		orientation and c_s	
$\mathbf{P}_{00}$	1	position covariance	4×4×*sizeof*(double)
$\mathbf{P}_{i0}$, where i≠0	n_T $= n_{c+}\ n_{e+}\ n_p$	position to feature cross-covariance	2×4×*sizeof*(double)
$\mathbf{P}_{ij}$, where i,j≠0	$n_T(n_T+1)/2$	feature to feature covariance	2×2×*sizeof*(double)

Omitting the low level details such as storage space for pulse samples and templates, the total memory requirement (in bytes) is

$$\text{Memory Requirement} = [4n_p + 3n_{po} + 20 + 12n_T + 2n_T^2] \times sizeof\,(\text{double}) + [2n_p + m_c + 2m_T] \times sizeof\,(\text{integer}) + n_T \quad (1)$$

where $m_T = m_c + m_e + m_p$

3.2 Processing Time

This subsection investigates the processing time required to process each type of new measurement. The notation for the *upper bound of order of growth* (Cormen, Leiserson and Rivest 1990), O(), is adopted to represent the processing time of the mapping algorithm. This is achieved by inspecting the overall structure of the algorithm and identifying the critical processes which are related to the number of existing features in a map, as shown in Table 2.

Table 2 : Simplified pseudo code for processing each measurement type and processing time.

New measurement	**Simplified pseudo code showing only the critical processes**	**Processing time**
Plane	**p1**. Find an adjacent and statistically collinear partial plane for fusion. Go to **a2** if found, go to **a1** if not found.	$O(n_p)$
Corner	**c1**. Find a statistically overlapping corner for fusion. Go to **a2** if found. Go to **c2** if not found.	$O(n_c)$
	c2. Find two partial planes which are adjacent to, and intersect at that corner. Go to **a2** if found, go to **a1** if not found.	$O(n_p^2)$
Edge	**e1**. Find a statistically overlapping edge for fusion. Go to **a2** if found. Go to **a1** if not found.	$O(n_e)$
All	**a1**. Create a new map feature state and covariance with all existing map features. End.	$O(n_T)$
	a2. Update all existing map feature states and covariance matrices with a residual vector. Go to **a3**.	$O(n_T^2)$
	a3. Project all endpoints of partial planes to the new states (ie. line parameters). Go to **a4**.	$O(n_p)$
	a4. Map Pruning, try to merge partial plane to partial plane, corner to corner, edge to edge and corner to two partial planes. End.	$O(n_c^2)+$ $O(n_e^2)+$ $O(n_c n_p^2)$

By analysing the flow of the above pseudo code, and merging some O() to their supersets (such as merging $O(n_p^2)$ and $O(n_p)$ to $O(n_T^2)$), three processing time possibilities are shown in Table 3.

Table 3 : Three possibilities of processing time estimation.

New measurement	Integration of new features	Fusion
Plane	$O(n_T)$	$O(n_T^2)+O(n_c n_p^2)$
Corner	$O(n_T)$ or $O(n_T) + O(n_p^2)$	
Edge	$O(n_T)$	

Further simplification of expressions will require a knowledge of the environment the robot explores, to enable us to predict the approximate number of map features and measurements. In the absence of this knowledge, any assumptions made regarding the likelihood of fusion and integration of new feature are groundless. For example, a worst case processing time is $O(n_T^2)+O(n_c n_p^2)$, obtained by assuming that data fusion dominates over integration of new features. This is a meaningless assumption because a good mapping robot should strive to seek new features by exploring unknown areas. A worst case processing time that hardly occurs does scant justice to an otherwise practical performance. For example, in long corridor type environments, integration of new features happens perceivably more frequently than data fusion. In section 6, the time taken to process all measurements at every navigation stop is plotted for each of the two experiments.

For operating systems that support virtual memory, disk access times need to be considered for large maps. The analysis involving virtual memory is beyond the scope of this paper.

4 The Map Matching Algorithm

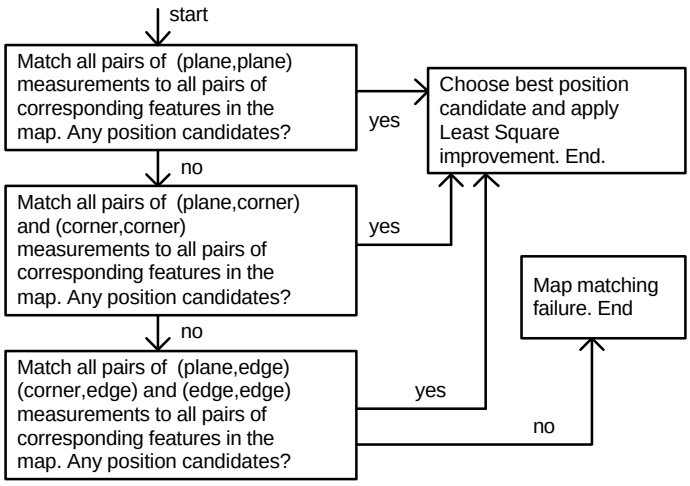

Figure 6 : Simplified map matching algorithm based on the reliability of features.

The details of the map matching scheme can be found in the (Chong and Kleeman 1996iv). To summarise, the robot attempts to localise by matching a set of new measurements to a map generated *a priori* with the map building method described in section 3. It requires the matching of at least two measurements to the corresponding features in the map to do a fix computation. By recognising the fact that planes produce the most reliable measurements, corners are more likely to produce phantom targets and edges have greater uncertainty, the algorithm saves time by arranging the search and fix computation according to the reliability of the features as shown in the simplified pseudo code in Figure 6. For each successful fix computation, the position candidate is recorded and a measure of discrepancy function is calculated to assess the overall quality of matching by accounting for other measurements not taking part in the fix computation. The measure of discrepancy has been designed to be robust against phantom targets and map imperfection. Eventually, the position candidate with the least discrepancy value is selected to be the best estimate and is

further enhanced by minimising a least square error formed from all matched feature pairs.

If the map is reasonably complete, the map matching algorithm can localise the robot without any approximate knowledge of its position with respect to the coordinate origin of the map. However, if the environment produces similar sensor data from many locations, such as a long corridor, the algorithm can fail. In practice an approximate knowledge of the robot position is available to guide the map matching process. In addition, by giving the robot the knowledge of the accuracy of the prior position information, it can prune the search space. Specifically, a search bound can be set around each measurement, such that only the map features of the same type that fall within the search bound become candidates for map matching. How this position accuracy information is generated for the map matching algorithm is described in the next section.

5 The New Local Mapping Algorithm

The fundamental rationale behind the new local mapping strategy is that if two features are far apart, the update of one feature will not significantly enhance the estimation of another feature through their correlation. Hence there is no reason to store their correlation matrix. Eliminating matrices of loose correlation not only saves memory, but improves processing speed.

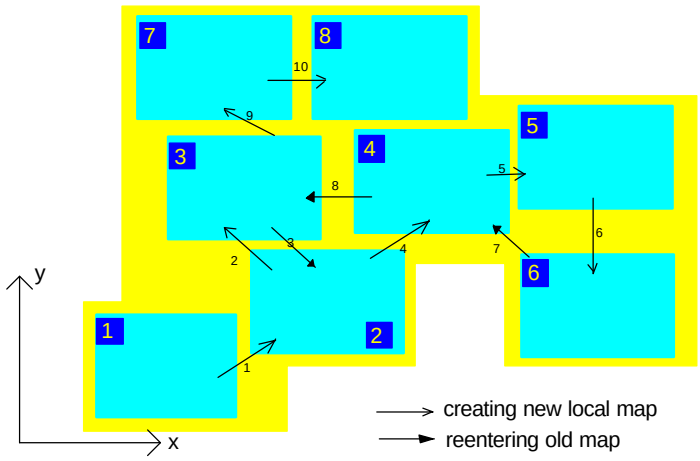

Figure 7 : Representation of a large environment as an interconnected set of local maps.

With the new strategy, the map and measurements retained in the memory are the same as the previous global approach, except that there are fewer since only objects in the current local map are present in the memory for processing at any one time. When the robot decides that there is a need for a new local map or suspects that an old local map has been revisited, the current local map is saved to disk. The memory is then cleared. If a new map is desired, the memory can be instantly put to use. If, on the other hand, an old map is to be revisited, it is retrieved from the disk. As a result only one small local map exists in memory at one time, leaving more memory for data processing. Throughout the navigation, the memory also constantly maintains, for each local map, a list of robot positions responsible for the formation of each of them. The purpose is to quickly check if the robot has revisited a previous local map before loading it into the memory and starting map match based validation. All positions are stored in a global coordinate system. Refer to Figure 7 for the two scenarios for creating a new local map and revisiting an old local map as the robot maps a large environment. Each local map can be viewed as a *node* of a graph, connected by *directed arcs* indicating the sequence of visits to local maps.

The complete local mapping algorithm is presented as a flowchart in Figure 8. The robot executes the same routine of sensing and mapping as in the global approach until the position covariance signals a switch to another local map. The criteria for the termination of the current local map is when the error covariance of the robot position exceeds a threshold, $\mathbf{Pr}_{TH}$ which is chosen so that the last few features in the current local map are not significantly correlated with the first few features. The robot then searches through the list of sensing positions to determine if it is re-visiting a local map. If one of the sensing positions is sufficiently close to the current position, the corresponding local map is loaded from disk and map matching is performed. As mentioned earlier, the current position information is used to prune the map matching search space. If matching is successful the robot continues by sensing and updating the loaded local map. Otherwise, another local map is tested until finally one, or none at all, is found. In the latter case, a new local map is created for further mapping. Each local map is assigned an index to indicate the time sequence of its generation.

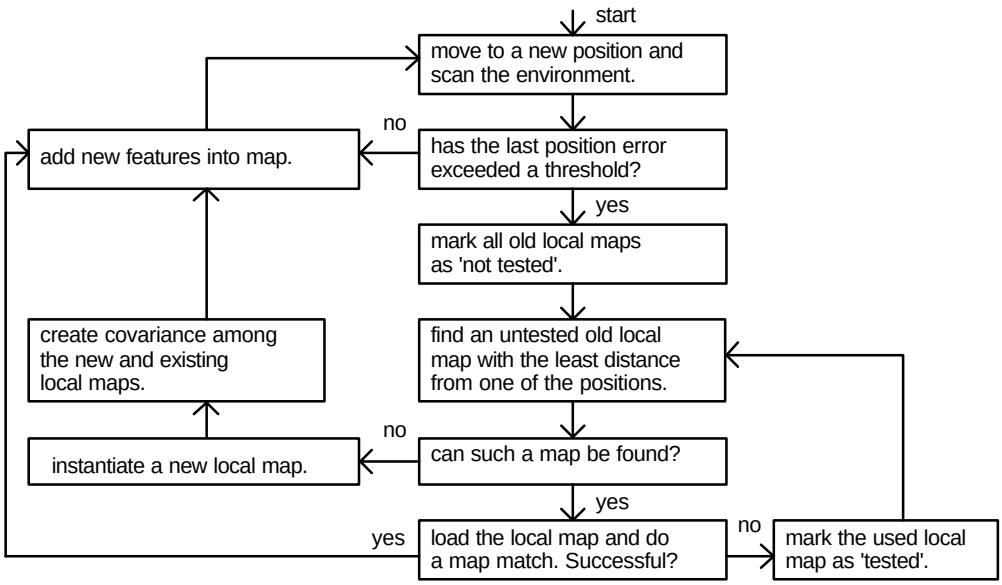

Figure 8 : Flowchart for the local mapping algorithm.

Map matching helps to alleviate the problem of error accumulation because the robot position, orientation and covariance are estimated from current sensor data and the loaded local map. Only the speed of sound estimate is carried over to the next local map. While the environment is still not perfectly mapped in a global sense, the robot can now practically remove all the error accumulated between visits to the same area. As shown later, this helps the robot to tackle challenging environments such as loops and long corridors.

5.1 Relative Covariance Between Local Maps

When a new local map is instantiated, its **relative covariances** with other local maps are created. The relative covariance, **Pr**, between two local maps is defined as the covariance between the robot positions at which the two local maps are first created. Relative covariance between local maps must be maintained in order to estimate the uncertainty of a robot position in one local map relative to another local map. This

information is used for statistically determining if the robot is revisiting a previous local map (see section 5.2 below), and for establishing a search bound for each measurement during map matching (see section 5.3 below).

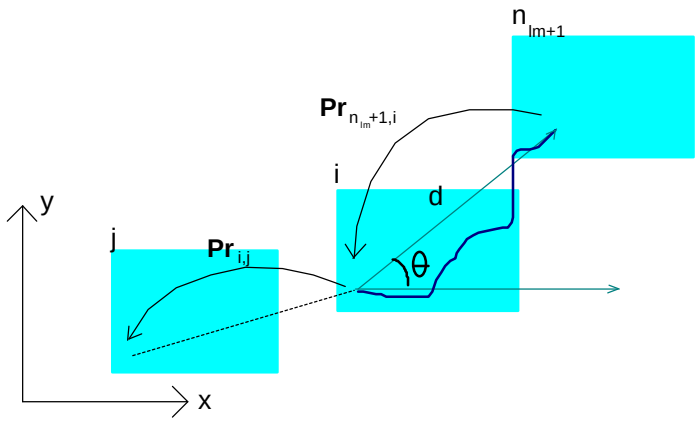

Figure 9 : Generating the relative covariance of a local map with respect to a previous local map

Referring to Figure 9, suppose that n_{lm} local maps already exist. The covariance of the position at which a new local map (local map $n_{lm}+1$) is to be created is $\mathbf{Pr}_{n_{lm}+1,i}$ with respect to the current local map i, and the covariance of the current local map with respect to a previous local map j ($j<i$) is $\mathbf{Pr}_{i,j}$, then the covariance of the new local map $n_{lm}+1$ with respect to the previous local map j is

$$\mathbf{Pr}_{n_{lm}+1,j} = \mathbf{Pr}_{n_{lm}+1,i} + \mathbf{T}(d,q)\mathbf{Pr}_{i,j}\mathbf{T}^{T}(d,q) \tag{2}$$

where

$$\mathbf{T}(d,q) = \begin{bmatrix} 1 & 0 & -d\sin(q) \\ 0 & 1 & d\cos(q) \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Equation (2) implicitly assumes that $\mathbf{Pr}_{i,j}$ and $\mathbf{Pr}_{n_{lm}+1,i}$ are uncorrelated. Attention is drawn to the specification that only the relative covariance with the previous local maps with indices less than the index of the current local map (ie. $j<i$) needs to be evaluated. Since local map indices are assigned in the order of their formation, this applies to any two local maps which are **monotonically linked** together, that is, one can reach local map *i* from local map *j* by traversing (following the directed arcs) a set of local maps with monotonically increasing indices. Refer to the example in Figure 7 again, the relative covariance between local map 3 and local map 4, $\mathbf{Pr}_{3,4}$ and $\mathbf{Pr}_{4,3}$ are not evaluated because one must pass through local map 2 to move from local map 3 to local map 4, and 2 is less than 3. Local map 2 is defined as the **common root** of local map 3 and local map 4. The common root of two monotonically linked local maps is the one with the smaller index. The reason for not evaluating the relative covariance between the non-monotonically linked local maps is that the uncertainty of any robot position relative to map 4 can be easily computed with a simple equation involving $\mathbf{Pr}_{3,2}$ and $\mathbf{Pr}_{4,2}$, as demonstrated in the next section. It also has a secondary advantage of reducing storage space.

5.2 Re-entrance of a Local Map

Suppose that the current local map has an index *i*, and the covariance of the latest robot position, p_i, relative to the current local map, is $\mathbf{P}_{pi,i}$. To check if a previous local map with index *j* is being revisited, the algorithm computes the distance between p_i and all robot positions responsible for the generation of local map *j*. If the shortest among the computed distance, d_{min}, is less than a threshold d_{TH}, local map *j* is loaded from disk for a trial map matching (section 5.3). However, if d_{min} is not within the distance threshold,

there is still a possibility that local map *j* is being revisited, if the uncertainty associated with this shortest distance is accounted for. We can begin by computing the covariance of p_i and the covariance of the position closest to it in local map *j*, p_j, with respect to their common root, *k*. With the covariance of the two positions, $\mathbf{P}_{pi,k}$ and $\mathbf{P}_{pj,k}$, the standard deviation of d_{min} can be subsequently determined.

The final problem is that the covariance of p_j relative to local map *j* is not stored in order to save memory. Nevertheless, one knows that it is bounded by $\mathbf{Pr}_{TH}$, the threshold which triggers a new map. Here $\mathbf{Pr}_{TH}$ is regarded as a worst case approximation to $\mathbf{P}_{pj,j}$.

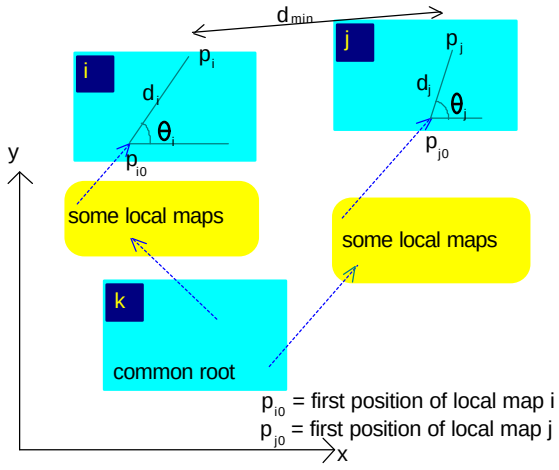

Figure 10 : Illustration of the computation of the standard deviation of d_{min}

By referring to Figure 10,

$$\mathbf{P}_{pi,k} = \mathbf{P}_{pi,i} + \begin{cases} \mathbf{0} & if \quad i = k \\ \mathbf{T}(d_i, q_i)\mathbf{Pr}_{i,k}\mathbf{T}^T(d_i, q_i) & if \quad i, j > k \end{cases} \tag{4}$$

$$\mathbf{P}_{pj,k} \leq \mathbf{Pr}_{TH} + \begin{cases} \mathbf{0} & if \quad j = k \\ \mathbf{T}(d_j, q_j)\mathbf{Pr}_{j,k}\mathbf{T}^T(d_j, q_j) & if \quad i, j > k \end{cases} \tag{5}$$

To compute the exact standard deviation of $d_{\min}$, the correlation between p_i and p_j with respect to local map *k* is required. Since this exact value is unimportant here, computational load can be reduced by approximating the standard deviation of the distance between p_i and p_j by

$$d_{d_{\min}} \approx \sqrt{\nabla_{pi} d_{\min} \mathbf{P}_{pi.k} \nabla_{pi} d_{\min}^T + \nabla_{pj} d_{\min} \mathbf{P}_{pj.k} \nabla_{pj} d_{\min}^T} \tag{6}$$

Local map *j* is declared **not revisited** at p_i if

$$d_{\min} > d_{TH} + 3d_{d_{\min}} \tag{7}$$

5.3 Search Bound for a New Measurement

Map matching follows the loading of local map *j*. As mentioned earlier, the map matching search space can be pruned by exploiting the approximate knowledge of the robot position provided by local map *i*. Just how many map features the algorithm should assign to each measurement depends on how accurately that prior position information is known. More specifically, the **search bound** of a new measurement should be set directly proportional to the accuracy of p_i relative to local map *j*.

Carrying forward the symbols defined in section 5.2, let the covariance of p_i relative to local map *j* be $\mathbf{P}_{pi,j}$. If local map *i* and local map *j* are monotonically linked,

$$\mathbf{P}_{pi,j} = \mathbf{P}_{pi,i} + \mathbf{T}(d_i, q_i)\mathbf{Pr}_{i,j}\mathbf{T}^T(d_i, q_i) \qquad if \qquad j < i \tag{8}$$

$$\mathbf{P}_{pi,j} = \mathbf{P}_{pi,i} + \mathbf{T}(d_j, q_j)\mathbf{Pr}_{j,i}\mathbf{T}^T(d_j, q_j) \qquad if \qquad j > i \tag{9}$$

if local map *i* and local map *j* are not monotonically linked and have a common root *k*,

$$\mathbf{P}_{pi,j} = \mathbf{P}_{pi,i} + \mathbf{T}(d_j, q_j)\mathbf{Pr}_{j,k}\mathbf{T}^T(d_j, q_j) + \mathbf{T}(d_i, q_i)\mathbf{Pr}_{i,k}\mathbf{T}^T(d_i, q_i) \tag{10}$$

for each measurement $\mathbf{m}=[r\ \rho]^T$ with covariance $\mathbf{Cov}(\mathbf{m})$ at p_i, which comprises the range and bearing to a classified reflector, relative to p_i. The Cartesian coordinates of the reflector are

$$\mathbf{x} = \begin{bmatrix} x_m & y_m \end{bmatrix}^T = \mathbf{H}(p_i, \mathbf{m}) \tag{11}$$

and its covariance relative to local map *j* is

$$\mathbf{Cov}(\mathbf{x}) = \nabla_{p_i}\mathbf{H}\mathbf{P}_{pi,j}\nabla_{p_i}\mathbf{H}^T + \nabla_m\mathbf{H}\mathbf{Cov}(\mathbf{m})\nabla_m\mathbf{H}^T \tag{12}$$

The search bound for **m** is defined as a circle centred at **x**, with a radius r_{BOUND} given by

$$r_{BOUND} = b\sqrt{trace(\mathbf{Cov}(\mathbf{x}))} \tag{13}$$

Since the position covariance is meant for establishing a search bound for a measurement, **not** map building, the value of the constant β is not critical. If the bound is so large that it captures more than one map feature for a measurement, the map matching algorithm will eliminate the unsuitable ones with its measure of discrepancy

function (Chong and Kleeman 1996iv). In the current implementation, β is set to 4. Similarly, whether the search bound is a circle or an ellipse is not critical due to the subsequent robust elimination process. The important thing is that the size of the search bound is **proportional** in size to the covariance of the robot position relative to the current local map. A circular search bound does nevertheless make geometrical computation a lot less complex.

5.4 Memory Requirements

The formula for assessing the memory requirements of the local mapping scheme is the same as that of the global mapping scheme (covered in section 3.1). However, the number of map elements stored in the memory is much less than that of the global mapping scheme. In addition to the map objects listed in Table 1, the global mapping scheme has to store the following objects:

Table 4 : Additional memory requirement for the local mapping strategy

Map objects	Quantity	RAM required by each object
robot positions responsible for each local map	$\sum_{i=1}^{n_{lm}} n_{po,i}$	3×*sizeof* (double)
covariance between local maps	$n_{lm}(n_{lm}+1)/2$	3×3×*sizeof* (double)

where n_{lm} is the number of local maps. $n_{po,i}$ is the number of robot positions in local map i. So the maximum additional memory required is

$$\text{Maximum additional Memory} = [4.5\, n_{lm}\,(n_{lm}+1) + 3\sum_{i=1}^{n_{lm}} n_{po,i}\,] \times sizeof(\text{double}) \qquad (14)$$

5.5 Processing Time

When processing a local map, the processing time estimation is the same as that described in section 3.2. Once again, since the number of map features to be processed is less than the local mapping strategy, the processing time is accordingly reduced. Nevertheless, the current implementation suffers the penalty of disk access times incurred when saving a local map to disk and when loading a local map from disk for trial map matching. This event occurs periodically, when the covariance of the position with respect to the current local map exceeds a preset threshold.

There are n_{lm} old local maps to check for the possibility of a revisit. For each local map, the algorithm checks through all n_{po} positions of each local map to check if the latest position is statistically close to the local map. If the possibility exists, the local map is loaded from disk to perform a map match. Once again, the processing time of the map matching process fluctuates tremendously depending on the current set of measurements and the state of map at the time, so an average case analysis and a worst case analysis are misleading for a practical strategy, as shown by the real experimental data.

6 Experimental Results

The local mapping algorithm has been tested in two large, real life environments. The paths were planned by a human operator so that instances of map matching could occur frequently enough to facilitate investigation. The paths were not designed to explore all regions of the test environment. The first environment is an approximately 50 meter long corridor. Figure 14 shows the raw sonar and odometry data generated from the

navigation along this corridor. The robot starts from the left end of the corridor, moves towards the right end, rotates 180° and travels back to a position very close to the starting position. The tiny circles along the path indicate sensing points, which are uniformly separated by 1 meter. The total journey is 93 meters. The tiny line segments are plane measurements; the dark pointy objects are corner measurements and the light pointy objects are edge measurements. Unknown objects are not displayed to avoid cluttering the diagram. the grid line spacing is 1 meter. Figure 15 shows the photographic views of the corridor. Typical landmarks the sonar sensor array pick up as map features are wall mouldings, table legs, wall discontinuities at doors, fire extinguishers, cupboards and concave corners between two walls and between a wall and furniture.

For this specific type of navigation, we expected that whilst the map built during the left-to-right part of the journey is inaccurate due to a lack of re-observation, the robot ought to be able to return to the left end of the corridor without producing an inconsistent map because it is roughly tracing the old path, observing the same landmarks in the reverse sequence. Therefore, as a preliminary step, the raw data is processed with the global mapping scheme. We found that the number of states quickly exceeded the pre-set limit of 200 before completing the navigation and virtual memory activities dominated the processor time. The global mapping scheme was processed a second time using the original raw data without the edge measurements. This time the navigation was completed before the number of states reached 200, and the result is shown in Figure 16. The left half of the corridor provides ample corner landmarks for localisation in the longitudinal direction (ie. parallel to the corridor). The right half of the corridor consists mainly of walls (planes) hence the error of the robot position in the

longitudinal direction grows more rapidly. We observed that when the robot starts making its return journey, a consistent map, albeit not strictly an accurate one with respect to the 'real' corridor, is built initially. Half way through, the robot encounters some poorly built features originated from some rather complex planar landmarks indicated on Figure 11 (in the feature map, the door in the middle has been mistakenly fused to the door on the right during the first pass), and fails to localise properly. After a series of poor localisations, fusion cannot occur subsequently despite the apparent re-observation of map features because the gross position errors force all validation tests to fail. Consequently, the left end of the corridor appears as two conflicting sections.

Figure 11 : Actual landmarks constituting the 'complex' area. In the feature map, the door in the middle has been fused to the door on the right.

When the local mapping scheme is employed, the same set of raw data yields six local maps for the complete navigation. They are superimposed together in Figure 17. All local maps are first generated when the robot moves from the left end to the right end of the corridor. After making the 180° turn, map matching is called on the way back to the left end of the corridor by reloading and building the local maps in the reverse order of their generation. According to Figure 14, if mapping had not been carried out, the robot would have been misled into believing that last position was about 2.5 meters from the starting position, and it was located in a new corridor below its starting corridor. After local mapping, the robot is about 0.3 meters from its starting position

(which was actually in agreement with physical observation) and the robot sees only one corridor throughout the navigation.

Table 5 compares the memory usage of the global map and each local map for this long corridor environment. Here *sizeof*(double) = 8 bytes and *sizeof*(integer) = 2 bytes. The average size of a local map is only 42 kbytes which is about 3% of the global map. According to (14), since n_{lm} = 6, and the total number of positions = 93, the local mapping scheme incurred an extra overhead of 3.7 kbytes, which adds a further 0.3%. If the global mapping scheme had been applied instead, the number of covariance matrices stored or to be processed in the memory, at the termination of the experiment, is roughly $279 \times 280/2 \approx 40{,}000$! Figure 18 compares the time to process all measurements collected at every sensing position for the global (that makes use of edge measurements) and local mapping algorithms. The growth rate of the processing time of the global approach is a much higher than that of the local approach. The global mapping process has to be terminated at the 55^{th} position because the number of features have reached 200 so the number of covariance matrices maintained is $200 \times 201/2 = 20{,}100$! The spikes on the global mapping graph in Figure 18 are proportional to the number of measurements fused into the map at various sensing points but some of which on the local mapping graph are caused by saving local maps to disk.

The second environment consists of a long corridor and a loop into a laboratory. Figure 19 shows the raw sonar and odometry data generated from the navigation around this environment. The robot starts at the position indicated with a ‘Start’ on Figure 19. It then moves right, turns right and enters a laboratory through a doorway, exits the lab at its starting position, moves left, makes a turn, travels back to its starting

position again, enters the lab and moves towards the other doorway. This time the robot moves in steps of 0.8 meters and the total journey is 82.4 meters. Figure 20 shows the photographic views of the corridor and the laboratory. Typical landmarks the sonar sensor array picks up as map features are similar to the first environment, but with more objects like boxes, trolleys and chair legs.

For the complete navigation, six local maps have been produced and they are superimposed in Figure 21. Attention is drawn to local map 4 when the robot exits the laboratory, revisits its starting position and moves left. One might wonder why the robot did not reload the first local map since it is re-entering it? The reason is that according to the algorithm, the robot switches to another map only when the position covariance is greater than $\mathbf{Pr}_{TH}$. By the time this happens, the robot is already quite a distance from the first local map, so it creates a new local map instead. In hindsight, the algorithm can be altered so that the robot checks for the possibility of revisiting an old region at **every** sensing point but this would incur extra processing delays undesirable for practical purposes.

Table 6 compares the memory usage of the global map and each local map for the second environment. The average size of a local map is only 41 kbytes which is about 4% of the global map. According to (14), since n_{lm} = 6, and the total number of positions = 104, the local mapping scheme incurred an extra overhead of 4 kbytes, which adds a further 0.35%. Figure 22 compares the time spent on processing all measurements collected at every sensing position for the global and local mapping algorithms. For this environment, the processing time of the global strategy is not drastically higher than the local strategy. In fact, at certain early positions, the processing time of the local strategy is actually higher due to saving local maps to disk.

Beginning at the 58th position, the global strategy gradually catches up with the local strategy. The global mapping process is terminated at the 71st position this time because the number of features have reached 200 at that point.

7 Limitations of the Algorithm

The algorithm contains several limitations which are discussed below.

If the environment has few landmarks, such as a long stretch of corridor with no wall mouldings or doorways, this mapping scheme may not work. In this case, a map matching error may occur, and a wrong position estimate might be established in the loaded local map. Consequently, the new map features subsequently sensed would be erroneously fused into the local map. Similarly, when searching for a previous local map, if there are inadequate landmarks (less than two) for map matching at that particular location, the robot will fail to recognise that it is re-visiting this map. Inevitably, a new local map which overlaps with a previous local map will be instantiated.

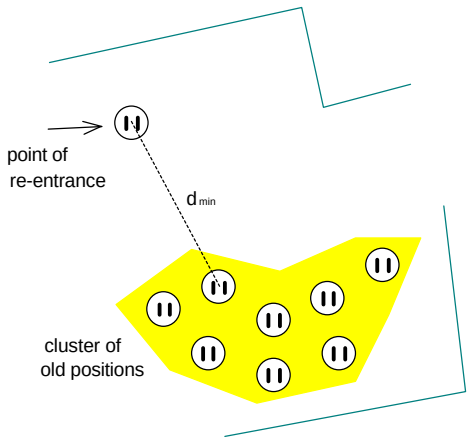

Figure 12 : The robot enters a previous local map but the algorithm fails to try map matching with this local map because d_{min} is too large.

The algorithm verifies the re-entrance of a previous local map based on checking if the robot current position is reasonably close to one of the positions from which the previous local map is produced. The decision is partly motivated by the fact that the robot is only allowed to travel a small step a time. Owing to the perceptive range of sonar, the spatial coverage of the local map is usually a lot larger than the cluster of positions associated with it. As a result, the robot might re-enter the previous local map from a location away from the cluster of positions, as depicted in Figure 12. With the current algorithm, the robot will reject the local map for map matching. The robot could have picked up this local map if the algorithm performs a map matching with **every** local map it possesses. This alternative is not implemented at this stage due to the computational burden. The consequence accompanying this decision is the possibility of overlapping local maps.

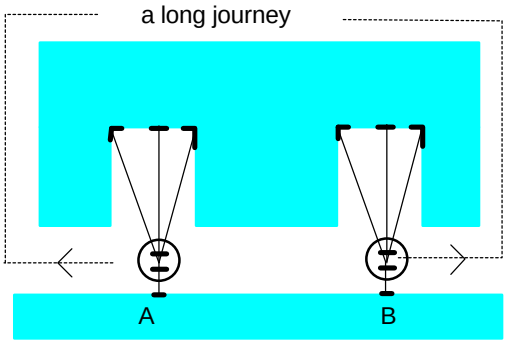

Figure 13 : The robot at position B has mistaken that it is back to location A after a long journey due to the similarity in observed landmarks at both positions, hence it loads the corresponding previous local map and performs a successful map match due to its large position covariance.

There will also be problem if the robot position covariance is too large when it is visiting a new area which is very close to an old area, and both areas are perceptually

very similar. As shown in Figure 13, if the path between position A and position B is long, the robot will be misled into loading the previous local map containing position A and perform a map match. The map matching will be successful due to the perceptual similarity between the surroundings of the two positions. On the other hand, if the robot seeks to backtrack its way back to position A with the current algorithm, it will have no problem returning to the real position A.

The reader should be aware that the local mapping strategy is devised to amend the global strategy which becomes sub-optimal[3] under the inevitable problems of nonlinearities, inaccurate knowledge of system noise variances and systematic errors. While the local mapping strategy is found to work successfully in a pragmatic sense, it is not a theoretically optimal strategy in the sense that when the robot performs fusion in a local map, the features in other local maps are not updated based on the relative covariance between the local maps. This is to improve efficiency and to curb the accumulated 'inevitable errors' from propagating to other local maps continuously. As a simple example, consider the scenario in which the robot creates a local map B after leaving a local map A. If the robot happens to re-enter local map A again and performs more sensing to obtain a better estimate of its position relative to local map A, this information is discarded if the robot enters local map B again. In other words, this improved estimate of position is not used to update the relative spatial position of local map B relative to local map A. It should be pointed out that in practice, the occurrence of these scenarios is rare.

[3] The global strategy is already sub-optimal due to the implementation of Relocation-Fusion (Moutarlier and Chatila 1989).

8 Conclusions

The new local mapping strategy has been tested in two large environments and has been shown to be memory friendly, time efficient, and capable of solving the problem of generating consistent maps when loops occur in the robot paths. As mentioned earlier, the robot was manually operated so that it covered wide enough an area before the battery went flat, to demonstrate the local mapping algorithm. As future work, the mapping work can be automated with a motion strategy such as that in (Chong and Kleeman 1996iii). On the fly navigation can be integrated with the stored local maps by predicting ahead which landmarks to sense to facilitate rapid localisation. For even larger environments, another layer of hierarchy can be mounted to group several adjacent local maps into a higher level local map. Another proposal worth researching is a way of fusing all local maps (including the overlapping ones), possibly using a least square approach, to generate one globally consistent map.

9 Acknowledgments

Mr. Greg Curmi's assistance in the design of the robot, the funding of a large ARC grant, the financial assistance provided by a MGS Scholarship and an OPRS Scholarship are all gratefully acknowledged.

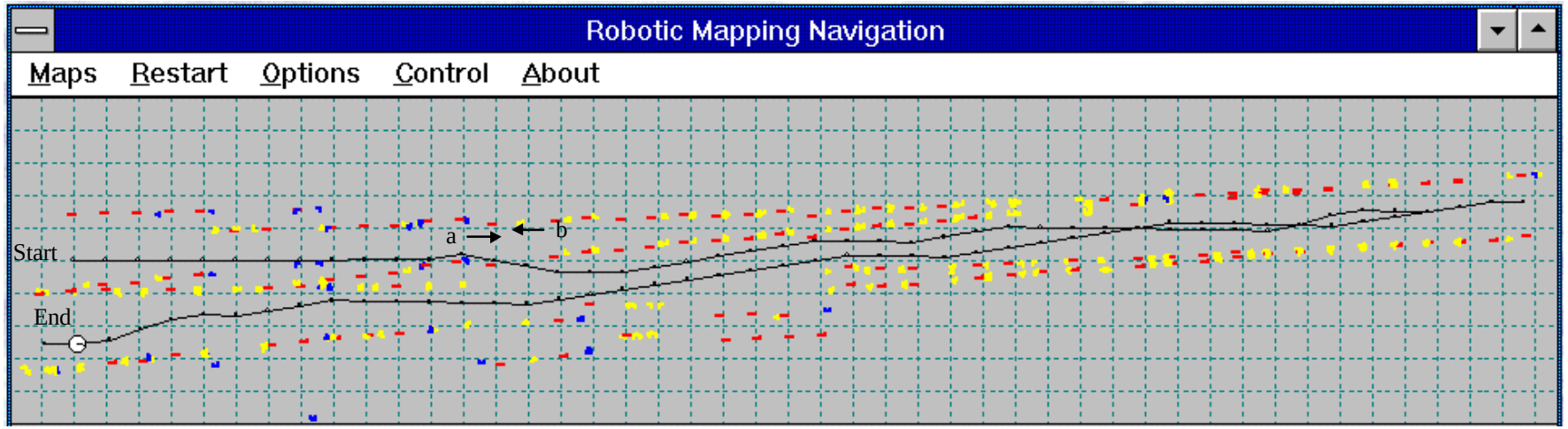

Figure 14 : Raw sonar data for the long corridor experiment (views 'a' and 'b' are shown in Figure 15)

(a)

(b)

Figure 15 : Various views of the long corridor

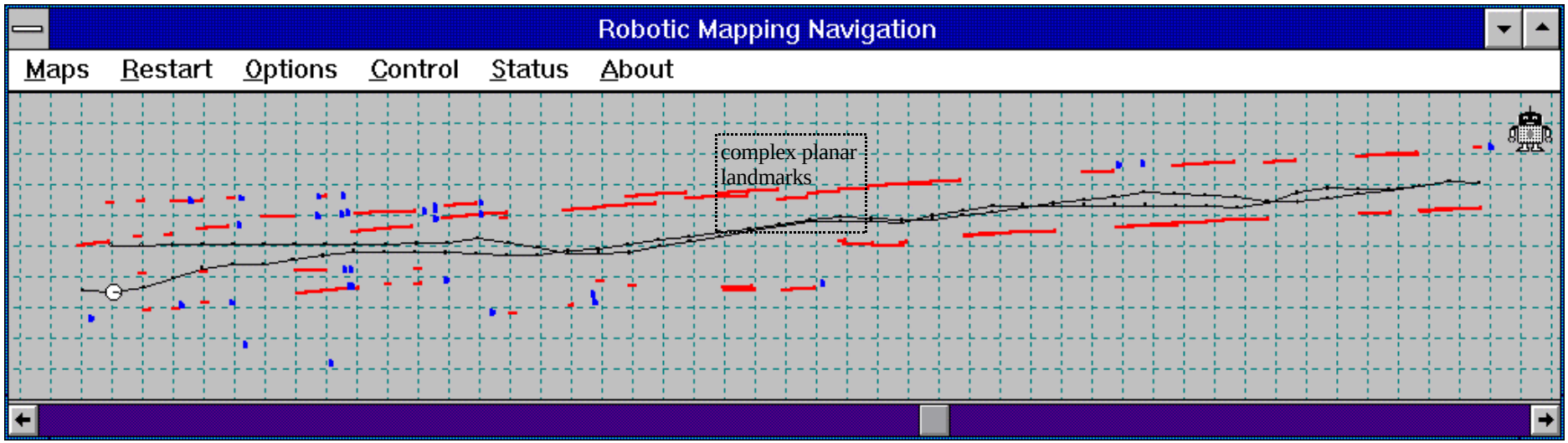

Figure 16 : Global mapping using plane and corner measurements only, for the long corridor experiment

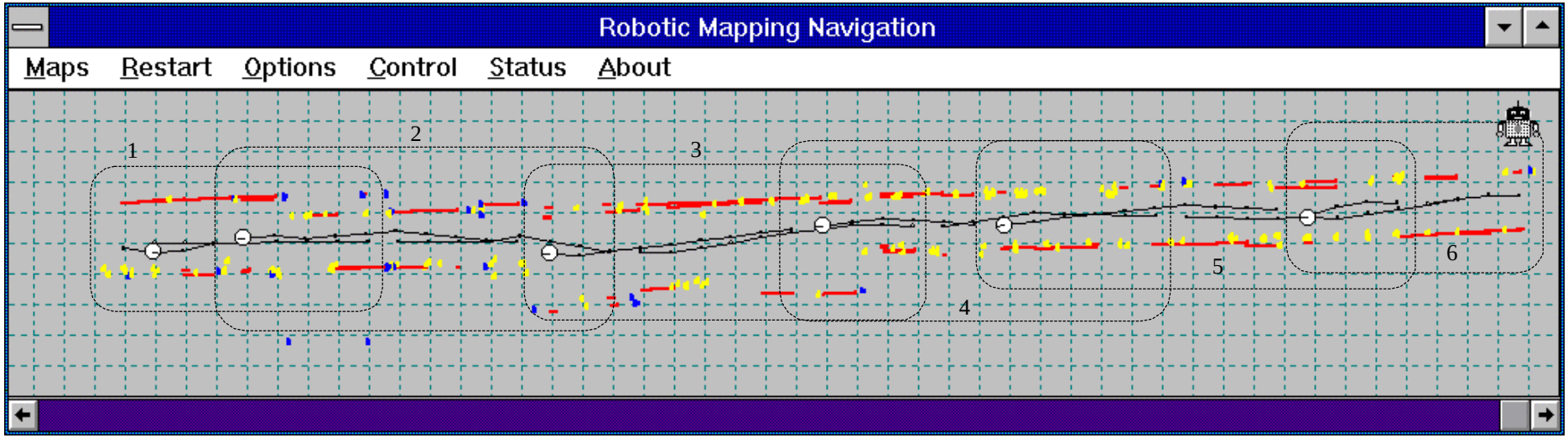

Figure 17 : Local mapping with all local maps (dashed boxes) being superimposed together

Table 5: Memory usage of the global map and every local map for the long corridor experiment.

	np	nc	ne	mp	mc	me	npo	nt	RAM (bytes)
Global Map	68	30	181	164	49	321	93	279	1279593
Local Map 1	8	7	26	22	7	48	13	41	31955
Local Map 2	10	14	25	27	29	43	18	49	44575
Local Map 3	15	5	34	46	9	53	19	54	53500
Local Map 4	12	1	40	24	1	66	14	53	51379
Local Map 5	15	2	39	28	2	76	18	56	57168
Local Map 6	8	1	17	17	1	35	11	26	14264

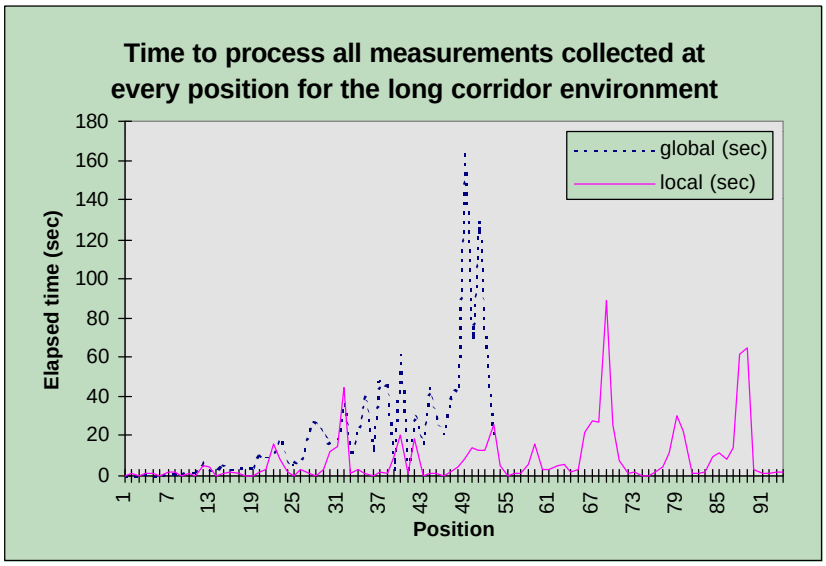

Figure 18 : Time to process all measurements collected at every position for the long corridor environment.

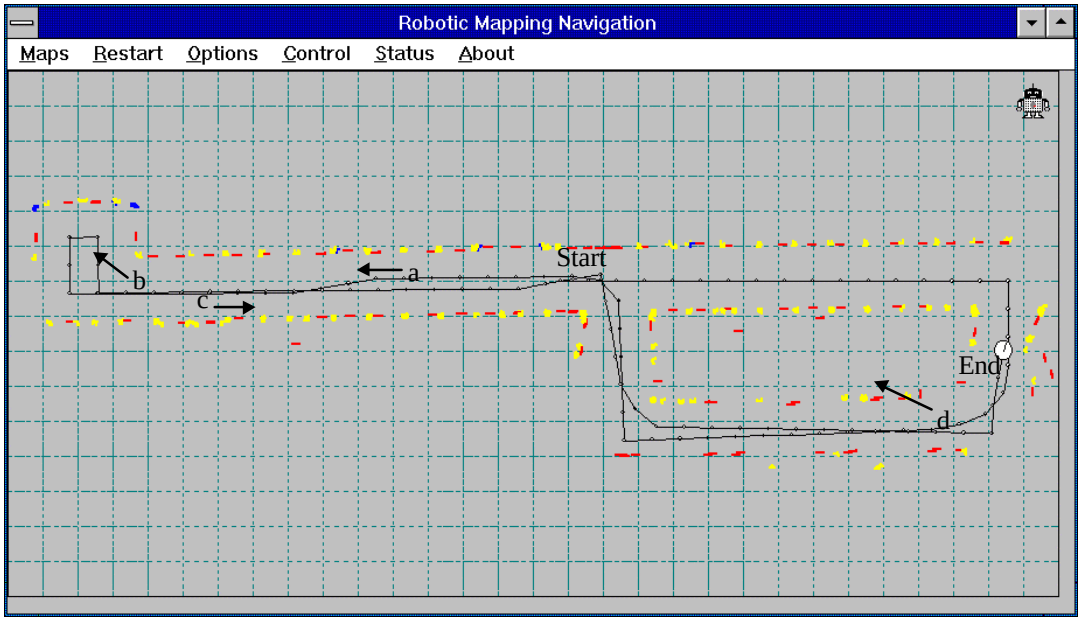

Figure 19 : Raw sonar data for the lab loop experiment (views 'a' - 'd' are in Figure 20).

Figure 20 : Various views of the lab loop.

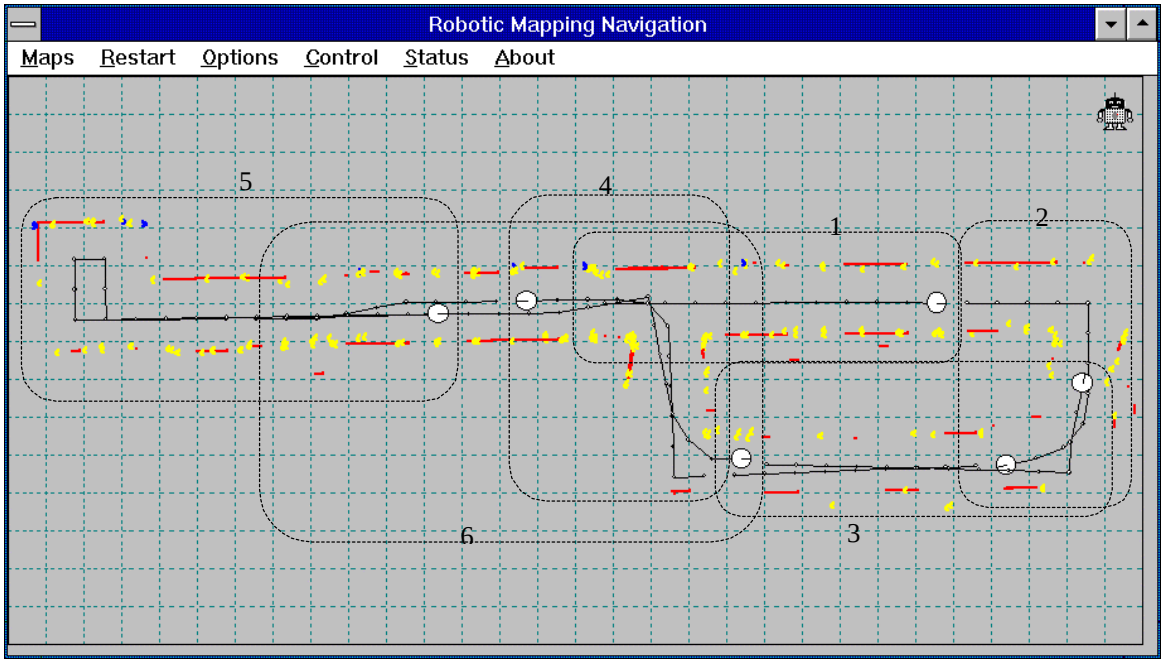

Figure 21 : Local mapping with all local maps (dashed boxes) superimposed.

Table 6: Memory usage of the global map and every local map for the lab loop environment.

	np	nc	ne	mp	mc	me	npo	nt	RAM (bytes)
Global Map	60	10	197	195	23	339	104	267	1173613
Local Map 1	7	1	37	22	1	55	12	45	37779
Local Map 2	9	0	23	20	0	39	12	32	20496
Local Map 3	10	0	18	23	0	29	23	28	16540
Local Map 4	7	2	21	21	3	33	12	30	18244
Local Map 5	15	3	52	53	3	96	23	70	87056
Local Map 6	12	4	46	56	16	87	22	62	69306

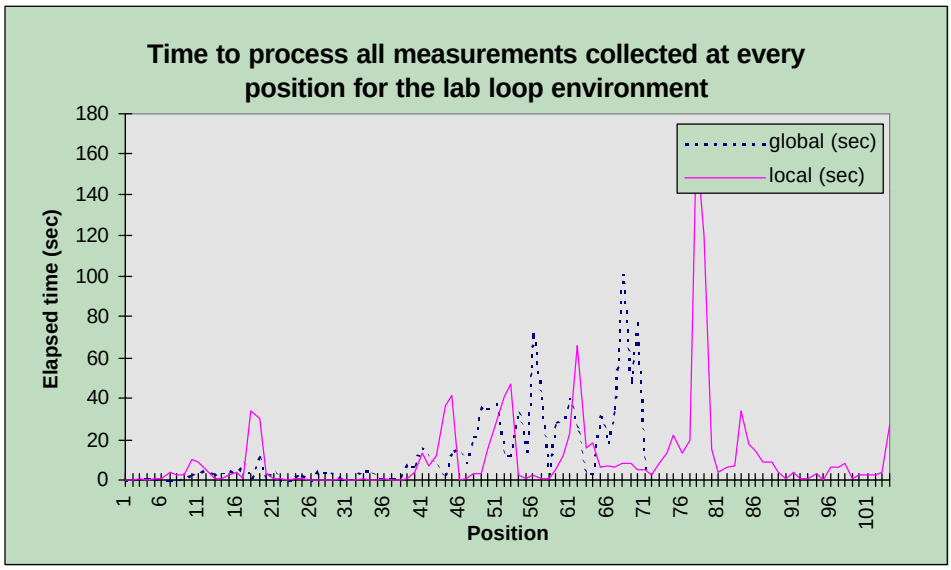

Figure 22 : Time to process all measurements collected at every position for the lab loop environment.

References

Asada, M., Fukui, Y. and Tsuji, S. 1990. Representing Global World of a Mobile Robot with Relational Local Maps. *IEEE Transaction on Systems, Man, and Cybernetics*. 20(6):1456-1461.

Bar-Shalom, Y. and Li, X.R. 1993. *Estimation and Tracking: Principles, Techniques and software*. Boston, London: Artech House Inc.

Cassandra, A.R., Kaebling, L. P. and Kurien, J. A 1996. Acting under Uncertainty: Discrete Bayesian Models for Mobile Robot Navigation, *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Osaka: pp. 963-972.

Chong, K. S. and Kleeman, L. 1996i. Accurate Odometry and Error Modelling for a Mobile Robot, Technical Report MECSE-1996-6. Department of Electrical and Computer Systems Engineering, Monash University.

Chong, K. S. and Kleeman, L. 1996ii. Sonar Feature Based Map Building for a Mobile Robot. Technical Report MECSE-1996-10. Department of Electrical and Computer Systems Engineering. Monash University.

Chong, K. S. and Kleeman, L. 1996iii. Indoor Exploration Using a Sonar Sensor Array: A Dual Representation Strategy, Technical Report MECSE-1996-12, Department of Electrical and Computer Systems Engineering, Monash University.

Chong, K. S. and Kleeman, L. 1996iv. Map Matching with Sonar Features. Technical Report MECSE-1996-13. Department of Electrical and Computer Systems Engineering. Monash University.

Chong, K. S. and Kleeman, L. 1997i. Accurate Odometry and Error Modelling for a Mobile Robot. *IEEE International Conference on Robotics and Automation*, Albuquerque: pp. 2783-2788.

Chong, K. S. and Kleeman, L. 1997ii. Sonar Feature Based Map Building for a Mobile Robot. *IEEE International Conference on Robotics and Automation*, Albuquerque: pp. 1700-1705.

Chong, K. S. and Kleeman, L. 1997iii. Indoor Exploration Using a Sonar Sensor Array: A Dual Representation Strategy. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Grenoble: pp. 676-682.

Cormen, T.H., Leiserson, C. E. and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press.

Firby, R.J., Christianson, D. and McDougal, T. 1992. Fast Local Mapping to Support Navigation and Object Localisation. *Proceedings of The International Society for Optical Engineering*. Vol. 1828. pp.344-352.

Jazwinski, A.H. 1970. *Stochastic Processes and Filtering Theory*. New York: Academic Press.

Julier, S., Uhlmann, J. and Durrant-Whyte, H. 1995. A New Approach for Filtering Non-linear Systems. *Proc 1995 American Control Conference*. Seattle, Washington: USA, pp. 1628-1632.

Kleeman, L. and Kuc, R. 1995 Mobile Robot Sonar for Target Localization and Classification, *Int. J. Robotics Res.* 14(4): 295-318.

Leonard, J.J. and Durrant-Whyte, H.F. 1991 (3-5 Nov) Simultaneous Map Building and Localisation for an Autonomous Robot. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*. pp. 1442-1447.

Malkin, P.K. and Addanki, S. 1990. LOGnets: A Hybrid Graph Spatial Representation for Robot Navigation, *AAA-90 Proceedings Eighth National Conference on Artificial Intelligence*. Vol. 2. pp.1045-1050.

Moutarlier, P. and Chatila, R. 1989. Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modeling. *5th International Symposium on Robotics Research:* Tokyo. pp.85-94.

Pearl, J. 1988. *Probabilistic Reasoning in Artificial Intelligence*, Morgan Kaufmann.