Sparse Local Submap Joining Filter for Building Large-Scale Maps

Shoudong Huang, *Member, IEEE*, Zhan Wang, and Gamini Dissanayake, *Member, IEEE*.

***Abstract*—This paper presents a novel local submap joining algorithm for building large-scale feature based maps: Sparse Local Submap Joining Filter (SLSJF). The input to the filter is a sequence of local submaps. Each local submap is represented in a coordinate frame defined by the robot pose at which the map is initiated. The local submap state vector consists of the positions of all the local features and the final robot pose within the submap. The output of the filter is a global map containing the global positions of all the features as well as all the robot start/end poses of the local submaps.**

Use of an Extended Information Filter (EIF) for fusing submaps makes the information matrix associated with SLSJF exactly sparse. The sparse structure together with a novel state vector and covariance submatrix recovery technique make the SLSJF computationally very efficient. The SLSJF is a canonical and efficient submap joining solution for large-scale Simultaneous Localization and Mapping (SLAM) problems that makes use of consistent local submaps generated by any reliable SLAM algorithm. The effectiveness and efficiency of the new algorithm is verified through computer simulations and experiments.

***Index Terms*—Simultaneous localization and mapping (SLAM), Extended Kalman Filter, Extended Information Filter, Map joining, Sparse matrix.**

I. Introduction

In the recent years, it has become evident that the Simultaneous Localization and Mapping (SLAM) problem can be efficiently solved by exploiting the sparseness of the information matrix or techniques from sparse graph and sparse linear algebra (see e.g. [1]-[5]). However, most of the methods based on sparse representation have focused on building a single large-scale map, resulting in the need to update a large map whenever a new observation is made.

Alternatively, local submap joining [6][7] provides an efficient way to build large-scale maps. In local submap joining, a sequence of small sized local submaps are built (e.g. using conventional Extended Kalman Filter (EKF) SLAM [8]) and then combined into a large-scale global map. During map joining [6], the state vector of the local submap is first transferred into the global coordinate frame. Common features present in both the local and global maps are identified and an EKF is used to enforce identity constraints to obtain the global map. The resulting map covariance matrix is fully correlated and thus the map fusion process is computationally demanding. Overall computational savings are achieved due to the fact that the frequency of global map updates is reduced.

This paper demonstrates that local submap joining can be achieved through the use of a sparse information filter. The proposed map joining filter, Sparse Local Submap Joining Filter (SLSJF), combines the advantages of the local submap joining algorithms and the sparse representation of SLAM to substantially reduce the computational cost of the global map construction.

The paper is organized as follows. Section II presents the overall structure of the SLSJF and demonstrates that the associated information matrix is exactly sparse. The SLSJF algorithm is described in detail in Section III. Section IV provides simulation and experiment results. Section V discusses some properties of the SLSJF and some related work. Section VI concludes the paper.

II. The overall structure of SLSJF

This section presents the overall structure of the SLSJF and explains why it results in a sparse representation.

A. The input and output of SLSJF

The input to the SLSJF is a sequence of local submaps constructed by some SLAM algorithm. Local maps [1] are denoted by

$$(\hat{X}^L, P^L) \tag{1}$$

where $\hat{X}^L$ (the superscript 'L' stands for the local map) is an estimate of the state vector

$$\begin{aligned} X^L &= (X_r^L, X_1^L, \cdots, X_n^L) \\ &= (x_r^L, y_r^L, \phi_r^L, x_1^L, y_1^L, \cdots, x_n^L, y_n^L) \end{aligned} \tag{2}$$

and P^L is the associated covariance matrix. The state vector X^L contains the robot final pose X_r^L (the subscript 'r' stands for the robot) and all the local feature positions $X_1^L, \cdots, X_n^L$, as typically generated by conventional EKF SLAM. The coordinate system of a local map is defined by the robot pose when the building of the local map is started, i.e. the robot starts at the coordinate origin of the local map.

It is assumed that the robot starts to build local map $k+1$ as soon as it finishes local map k. Therefore the robot end pose

Manuscript received June 5, 2007; revised ?? ??, 2008. This paper was recommended for publication by Associate Editor ???? and Editor L. Parker upon evaluation of the reviewers comments. This work was supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government. The material in this paper was partially presented at 2008 IEEE International Conference on Robotics and Automation, Pasadena, California, USA, May 19-23, 2008.

The authors are with ARC Centre of Excellence for Autonomous Systems (CAS), Faculty of Engineering, University of Technology, Sydney, Australia (email: sdhuang@eng.uts.edu.au; zhwang@eng.uts.edu.au; gdissa@eng.uts.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier ???

[1] In this paper, "local submap" is sometimes simply called "local map".

of local map k (defined as the global position of the last robot pose when building local map k) is the same as the robot start pose of local map $k+1$ (Fig. 1).

The output of SLSJF is a global map. The global map state vector contains all the feature positions and all the robot end poses of the local maps (see Fig. 1).

B. Why can local map joining have sparse representation?

The reason why SLSJF can be developed is that the information contained in each local map is the relative position information about some "nearby objects" — the features and the robot start/end poses involved in the local map.

By including all the objects (all the features and all the robot start/end poses) in the global map state vector, the local map joining problem becomes a large-scale estimation problem with only "local" information (similar to Smooth and Mapping (SAM) [2] and full SLAM [5]). When Extended Information Filter (EIF) is used to solve the estimation problem, a non-zero off-diagonal element of the information matrix (a "link" between the two related objects) occurs only when the two objects are within the same local map [2]. Since the size of each local map is limited, any object will only have link with its "nearby objects" no matter how many (overlapping) local maps are fused (Fig. 1). This results in an exactly sparse information matrix.

Since all the objects involved in the local maps are included in the global state vector, no marginalization is required in the map joining process and thus the information matrix will stay exactly sparse all the time. Because most of the robot poses are marginalized out during the local map building process, the dimension of the global state vector is much less than that of SAM [2] and full SLAM [5].

C. The overall structure of SLSJF

SLSJF fuses the local maps sequentially to build a global map, in a manner similar to [6][7], using the structure presented in Algorithm 1.

Algorithm 1 Overall structure of SLSJF

Require: A sequence of *local maps*: each local map contains a state vector estimate and a covariance matrix

1: Set local map 1 as the *global map*
2: For $k = 2 : p$ (p is the total number of local maps), fuse *local map* k into the *global map*
3: End

III. The SLSJF algorithm

This section describes the various steps of SLSJF algorithm, including global map initialization and update, reordering of the global state vector, state vector and covariance submatrix recovery, and data association.

[2] An off-diagonal element of the information matrix is exactly zero if the two related variables are conditionally independent given all the other variables (see e.g. [9] for a proof). In local map joining, two objects are conditionally independent unless they are involved in the same local map.

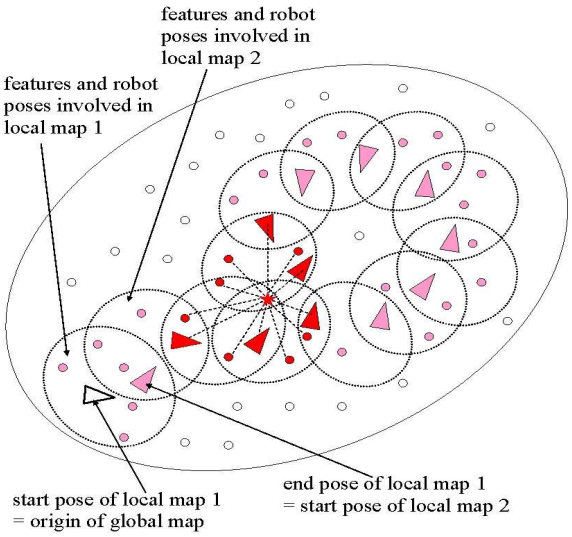


Fig. 1. Structure of SLSJF: Small ellipses indicate the objects involved in the local maps. Each object (e.g. the feature $\star$) is only linked to its "nearby objects" (features and robot poses that share the same local map with it). The final global map state vector contains the locations of all the shaded objects.

A. State vector of the global map

The state vector of the global map contains the global positions of all features and all the robot end poses of the local maps. For convenience, the origin of the global map is chosen to be the same as the origin of local map 1 (Fig. 1).

After local maps 1 to k are fused into the global map, the global state vector is denoted as $X^G(k)$ (the superscript 'G' stands for the global map) and is given by

$$\begin{aligned} & X^G(k) \\ = \; & (X^G_1, \cdots, X^G_{n_1}, X^G_{1e}, \\ & X^G_{n_1+1}, \cdots, X^G_{n_1+n_2}, X^G_{2e}, \\ & \cdots\cdots \\ & X^G_{n_1+\cdots+n_{k-1}+1}, \cdots, X^G_{n_1+\cdots+n_{k-1}+n_k}, X^G_{ke}) \end{aligned} \tag{3}$$

where $X^G_1, \cdots, X^G_{n_1}$ are the global positions of the features in local map 1; $X^G_{n_1+1}, \cdots, X^G_{n_1+n_2}$ are the global positions of those features in local map 2 but not in local map 1; $X^G_{n_1+\cdots+n_{k-1}+1}, \cdots, X^G_{n_1+\cdots+n_{k-1}+n_k}$ are the global positions of the features in local map k but not in local maps 1 to $k-1$. $X^G_{ie} = (x^G_{ie}, y^G_{ie}, \phi^G_{ie})$ $(1 \le i \le k)$ is the global position of the robot end pose of local map i, which is also the robot start pose of local map $i+1$. Here the subscript 'e' stands for robot 'end pose'.

In the information filter framework, an information vector $i(k)$ and an information matrix $I(k)$ is used for map fusion. The relationship between state vector estimate $\hat{X}^G(k)$ and the corresponding covariance matrix $P(k)$ and $i(k)$, $I(k)$ is ([5])

$$I(k)\hat{X}^G(k) = i(k), \quad P(k) = I(k)^{-1}. \tag{4}$$

As $I(k)$ is an exactly sparse matrix, it can be stored and computed efficiently. The state vector estimate $\hat{X}^G(k)$ is recovered after fusing each local map by solving the sparse

linear equation (the first equation in (4)). The whole dense matrix $P(k)$ is neither computed nor stored in SLSJF. Small parts of $P(k)$ required for data association are computed by solving a set of sparse linear equations, as outlined in Section III-C3.

When fusing local map $k+1$ into the global map, the features that are present in local map $k+1$ but have not yet been included in the global map, $X^G_{n_1+\cdots+n_k+1}, \cdots, X^G_{n_1+\cdots+n_k+n_{k+1}}$, together with the robot end pose of local map $k+1$, $X^G_{(k+1)e}$, are added into the global state vector $X^G(k)$ in (3) to form the new state vector $X^G(k+1)$.

B. Steps of local map fusion

The steps used in fusing local map $k+1$ into the global map are listed in Algorithm 2.

Algorithm 2 Fuse local map $k+1$ into global map

Require: *global map* and *local map* $k+1$

1: Data association
2: Initialize the new features and $X^G_{(k+1)e}$ in the global map
3: Update the global map
4: Reorder the global map state vector when necessary
5: Compute the Cholesky Factorization of $I(k+1)$
6: Recover the global map state estimate $\hat{X}^G(k+1)$

C. Data Association

Data association refers to finding the features in local map $k+1$ that are already included in the global map and their corresponding indices in the global state vector. This is an essential step in any practically deployable SLAM algorithm, yet is often neglected in many of the sparse information filter based SLAM algorithms published in the literature.

Data association is a challenge problem in EIF SLAM, if only the geometric relationships among features present in the global and local maps are available. How this can be efficiently achieved in SLSJF is described in the following.

Algorithm 3 Data association between local map $k+1$ and the global map

Require: *global map* and *local map* $k+1$

1: Determine the set of potentially overlapping local maps
2: Find the set of potentially matched features
3: Recover the covariance submatrix associated with X^G_{ke} and the potentially matched features
4: Use a statistical data association method to find the match

1) The set of potentially overlapping local maps: Local map i cannot overlap with local map $k+1$ if the distance between the origins of the two maps in the global coordinate frame, is larger than the sum of the two local map radii plus the possible estimation error. The radius of a local map is defined as the maximal distance from the local map features to its origin. Fig. 2 illustrates the idea. Note that the location estimate of the origin of local map i is $\hat{X}^G_{(i-1)e}$ (for $2 \le i \le k$), while that of local map 1 is $(0,0,0)$.

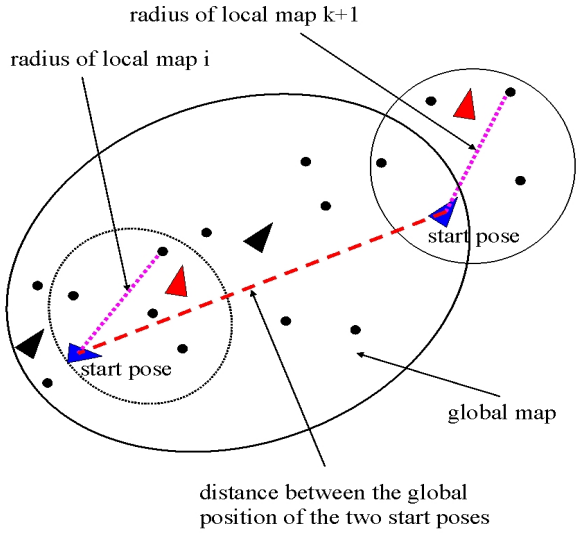


Fig. 2. Finding the potentially overlapping local maps: If the distance between the global position of robot start pose in local map i and the global position of robot start pose in local map $k+1$ is larger than the sum of the two local map radii, then the two local maps cannot overlap.

2) The set of potentially matched features: A feature from potentially overlapping local maps is added to a potentially matched feature list, if the distance from it to $\hat{X}^G_{ke}$ is smaller than the radius of local map $k+1$ plus the maximal possible estimation error. This list is further simplified by removing any member that is located further than a predetermined threshold distance from all features in local map $k+1$.

3) Covariance submatrix associated with X^G_{ke} and all potentially matched features: The covariance submatrix can be obtained by first computing the corresponding columns of the covariance matrix $P(k)$ and then extracting the desired rows.

Using (4), the l-th column of the covariance matrix $P(k)$, P_l, can be obtained by solving the sparse linear equation [10]

$$I(k)P_l = e_l \tag{5}$$

where

$$e_l = [\overbrace{0, \cdots, 0}^{l-1}, 1, , 0 \cdots, 0]^T. \tag{6}$$

Since the Cholesky factorization of $I(k)$, L_k, is a triangular matrix satisfying $L_k L_k^T = I(k)$, the sparse linear equations (5) can be solved efficiently by first solving $L_k q = e_l$ and then solving $L_k^T P_l = q$ [2][11]. Note that the Cholesky factorization L_k is already available from Step 5 of Algorithm 2 when fusing local map k into the global map, as described in Section III-G.

4) Feature matching: Since both the state estimates and the covariance matrices of the potentially matched features are available, any statistical data association algorithm (such as the simple Nearest Neighbor method [8] or the more robust Joint Compatibility Test with branch and bound technique [12]) can be used to find the matching features.

D. Initialize the new features and $X^G_{(k+1)e}$ in the global map

The initial values of the global positions of all unmatched features and the robot end pose of local map $k+1$ are computed (using $\hat{X}^G_{ke}$ and the local map state estimate) and inserted to $\hat{X}^G(k)$ to form a new state vector estimate $\hat{X}^G(k)$. The dimensions of $i(k)$, $I(k)$ and L_k are increased by adding zeros to form a new information vector $i(k)$, a new information matrix $I(k)$, and the corresponding Cholesky factorization L_k.

E. Update the global map

Suppose local map $k+1$ is given by (1). Since the local map provides a consistent estimate of the relative positions from robot start pose to the local features and the robot end pose, this map can be treated as an observation of the true relative positions with a zero-mean Gaussian observation noise whose covariance matrix is P^L.

To state it clearly, suppose the data association result is $X^L_1 \leftrightarrow X^G_{i1}, \cdots, X^L_n \leftrightarrow X^G_{in}$ (including both old and new features). Then the local map state estimate $\hat{X}^L$ can be regarded as an observation of the true relative positions from X^G_{ke} to $X^G_{i1}, \cdots, X^G_{in}, X^G_{(k+1)e}$. That is,

$$z_{map} = \hat{X}^L = H_{map}(X^G) + w_{map} \tag{7}$$

where $H_{map}(X^G)$ is the vector of relative positions given by

$$\begin{pmatrix} (x^G_{(k+1)e} - x^G_{ke})\cos\phi^G_{ke} + (y^G_{(k+1)e} - y^G_{ke})\sin\phi^G_{ke} \\ (y^G_{(k+1)e} - y^G_{ke})\cos\phi^G_{ke} - (x^G_{(k+1)e} - x^G_{ke})\sin\phi^G_{ke} \\ \phi^G_{(k+1)e} - \phi^G_{ke} \\ (x^G_{i1} - x^G_{ke})\cos\phi^G_{ke} + (y^G_{i1} - y^G_{ke})\sin\phi^G_{ke} \\ (y^G_{i1} - y^G_{ke})\cos\phi^G_{ke} - (x^G_{i1} - x^G_{ke})\sin\phi^G_{ke} \\ \vdots \\ (x^G_{in} - x^G_{ke})\cos\phi^G_{ke} + (y^G_{in} - y^G_{ke})\sin\phi^G_{ke} \\ (y^G_{in} - y^G_{ke})\cos\phi^G_{ke} - (x^G_{in} - x^G_{ke})\sin\phi^G_{ke} \end{pmatrix}$$

and w_{map} is the zero-mean Gaussian "observation noise" whose covariance matrix is P^L.

The "observation" z_{map} can now be used to update the information vector and the information matrix as follows:

$$\begin{aligned} I(k+1) &= I(k) + \nabla H^T_{map}(P^L)^{-1}\nabla H_{map} \\ i(k+1) &= i(k) + \nabla H^T_{map}(P^L)^{-1}[z_{map} \\ &\quad - H_{map}(\hat{X}^G(k)) + \nabla H_{map}\hat{X}^G(k)] \end{aligned} \tag{8}$$

where ∇H_{map} is the Jacobian of the function H_{map} with respect to $X^G(k)$ evaluated at $\hat{X}^G(k)$.

Since $z_{map} = \hat{X}^L$ only involves two robot poses $X^G_{ke}, X^G_{(k+1)e}$ and some local features (a small fraction of the total features in the global map), the matrix $\nabla H^T_{map}(P^L)^{-1}\nabla H_{map}$ in (8) and the information matrix $I(k+1)$ are both exactly sparse.

F. Reorder the global map state vector when necessary

The purpose of reordering the global state vector is to make the computation of Cholesky factorization (Section III-G), the state vector recovery (Section III-H), and the covariance submatrix recovery (Section III-C3) more efficient. Many different strategies for reordering are available. The strategy proposed here is a combination of the Approximately Minimal Degree (AMD) reordering [2][13] and the reordering based on distances [4].

Whether to reorder the global map state vector or not depends on where the features in local map $k+1$ are located within the global state vector. If all of the features in local map $k+1$ are present within the n_0 elements from the bottom of the global state vector [3], then the state vector is left unchanged. If this condition is violated, which happens only when closing a large loop, then the state vector is reordered.

The state vector is reordered using the following process. The robot pose $X^G_{(k+1)e}$ and the features that are within distance d_0 [4] to $\hat{X}^G_{(k+1)e}$ are placed at the bottom part of the state vector. Their order is determined based on the distances from them to $\hat{X}^G_{(k+1)e}$. The smaller the distance, the closer the position to the bottom. All the other robot poses and features are placed in the upper part of the state vector, they are reordered based on AMD.

The major advantage of reordering by AMD is that the number of fill-ins in Cholesky factorization will be reduced. The major advantage of reordering the nearby features based on distances is that once the reordering is performed, another reordering will not be required for the next few local map fusion. This is because the robot cannot observe features that are not located in the bottom part of the state vector until it travels a certain distance.

Once the state vector is reordered, the corresponding information matrix $I(k+1)$ and information vector $i(k+1)$ are reordered accordingly. For notational simplicity, they are still denoted as $I(k+1)$ and $i(k+1)$. Note that the Cholesky factorization of the reordered $I(k+1)$ cannot be easily obtained from L_k.

G. Compute the Cholesky factorization of $I(k+1)$

The method used to compute the Cholesky factorization of $I(k+1)$ depends on whether the global state vector was reordered in Section III-F or not.

Case (i). If the global state vector was not reordered in Section III-F, then the Cholesky factorization of $I(k)$ (available from Step 5 of Algorithm 2 when fusing local map k) is used to construct the Cholesky factorization of $I(k+1)$ as follows.

By (8), the relation between $I(k+1)$ and $I(k)$ is

$$I(k+1) = I(k) + \begin{bmatrix} 0 & 0 \\ 0 & \Omega \end{bmatrix} \tag{9}$$

where the upper-left element in Ω is non-zero. Here Ω is a symmetric matrix determined by the term $\nabla H^T_{map}(P^L)^{-1}\nabla H_{map}$ in (8). Its dimension is less than n_0 since otherwise the state vector would have been reordered.

[3]The threshold n_0 needs to be properly chosen in order to make the SLSJF algorithm efficient. A smaller n_0 will make the incremental Cholesky factorization step (Case (i) in Section III-G) more efficient but will also increase the total number of reordering and the direct Cholesky factorization operations (Case (ii) in Section III-G). As a rule of thumb, n_0 can be chosen to be around one tenth of the dimension of the global state vector.

[4]The threshold d_0 is related to the parameter n_0; it also depends on the feature density of the environment. The guideline is that the number of features that are within distance d_0 to $\hat{X}^G_{(k+1)e}$ is around half of n_0.

Let $I(k)$ and its Cholesky factorization L_k (a lower triangular matrix) be partitioned according to (9) as

$$I(k) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22} \end{bmatrix}, \quad L_k = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}. \quad (10)$$

According to (9) and (10), $I(k+1)$ can be expressed by

$$I(k+1) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}^{k+1} \end{bmatrix} = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22} + \Omega \end{bmatrix}. \quad (11)$$

By Lemma 1 in the Appendix of [4], the Cholesky factorization of $I(k+1)$ can be obtained by

$$L_{k+1} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22}^{k+1} \end{bmatrix} \quad (12)$$

where L_{22}^{k+1} is the Cholesky factorization of the low dimensional matrix $\Omega + L_{22}L_{22}^T = I_{22}^{k+1} - L_{21}L_{21}^T$.

Computing L_{k+1} by (12) is much more efficient than directly computing the Cholesky factorization of the high dimensional matrix $I(k+1)$.

Case (ii). If the global state vector has been reordered in Section III-F, then the Cholesky factorization of $I(k)$ cannot be used to construct the Cholesky factorization of $I(k+1)$. In this case, a direct Cholesky factorization of $I(k+1)$ is performed to obtain L_{k+1}.

Since the reordering only happens occasionally, Case (i) occurs most of the time.

H. State vector recovery

Because the global map is maintained as an information vector and an information matrix, the global state estimate $\hat{X}^G(k+1)$ is not directly available. Using (4), the state vector estimate $\hat{X}^G(k+1)$ can be recovered by solving the sparse linear equation

$$I(k+1)\hat{X}^G(k+1) = i(k+1). \quad (13)$$

The Cholesky factorization L_{k+1} computed in Section III-G is used to solve the sparse linear equation. Since $L_{k+1}L_{k+1}^T = I(k+1)$, the sparse linear equation (13) can be solved efficiently by solving $L_{k+1}Y = i(k+1)$ and $L_{k+1}^T\hat{X}^G(k+1) = Y$.

IV. Simulation and experiment results

In this section, simulation and experiment results are given to illustrate the accuracy and efficiency of SLSJF.

A. Simulation results

The $150 \times 150m^2$ simulation environment used contains 2500 features arranged in uniformly spaced rows and columns. The robot started from the left bottom corner of the square and followed a random trajectory as shown in Fig. 3(a). A sensor with a field of view of 180 degrees and a range of 6 meters (the small semi-circle seen near the bottom in Fig. 3(a)) was simulated to generate relative range and bearing measurements between the robot and the features. There were 27924 robot poses in total and 170846 measurements were made from the robot poses. The robot observed 2270 features in total and most of them were observed a number of times.

Six hundred small sized local maps were built by conventional EKF SLAM using the odometry and measurement information. Each local map contains around 10 features. Fig. 3(a) shows the global map generated by fusing all the 600 local maps using SLSJF. The data association in SLSJF was performed using Nearest Neighbor method [8]. The global map was superimposed with the global map generated by fusing the 600 local maps using EKF sequential map joining [6][7] and the map generated by a single EKF SLAM. Close examination (e.g. Fig. 3(b)) shows that the feature position estimates computed by the three methods are all consistent. The feature position estimates of SLSJF and EKF sequential map joining are almost identical.

Fig. 3(c) shows the errors and 2σ bounds of the estimates of the 600 robot end poses obtained using the three methods. It is clear that the estimates are all consistent. It should be noted that in SLSJF, the robot end poses are included in the global state vector and are continuously updated. Therefore the error and 2σ bounds of SLSJF are smaller than that of EKF sequential map joining and EKF SLAM where the robot poses except the most recent are not included in the state vector (hence are not updated).

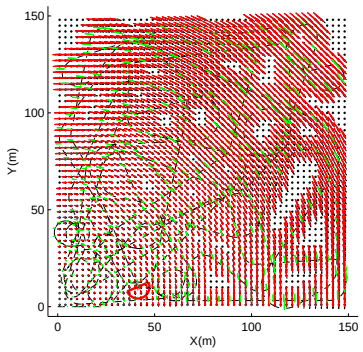
Fig. 3(d) shows all the non-zero elements of the sparse information matrix obtained by SLSJF in black. Fig. 3(e) shows the CPU time [5] required for the local map fusion using SLSJF and EKF sequential map joining. The total time for fusing all the 600 local maps is 145 seconds for SLSJF and 7306 seconds for EKF sequential map joining (building the 600 local maps takes 95 seconds, it takes conventional EKF SLAM more than 15 hours to finish the map). Table I presents the detailed processing time for the two map joining algorithms. In SLSJF, the major computation cost is due to "data association" which includes the time for covariance submatrix recovery. The "others" including reordering of the state vector, Cholesky factorization and state vector recovery also take significant time. On the other hand, "global map update" uses most of the computation time in EKF sequential map joining.

Fig. 3(f) compares the CPU time of SLSJF with the proposed reordering strategy and that of SLSJF with the AMD-only reordering [2][13] (for the proposed reordering, the parameters $n_0 = 400$ and $d_0 = 15$, for the AMD-only reordering, the reordering is performed after fusing every 5 local maps, the parameters are chosen such that both algorithms have their best performance). The performance of the two reordering algorithms are very similar, presumably due to the fact that the MATLAB implementation of AMD algorithm is very efficient.

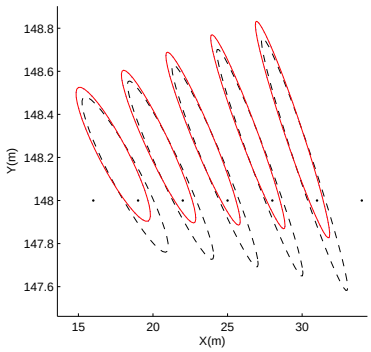
B. Experimental results

SLSJF was also applied to the Victoria Park data set which was first used in [14]. Neither ground truth nor noise parameters are available for this data set. Published results for the vehicle trajectory and uncertainty estimates vary

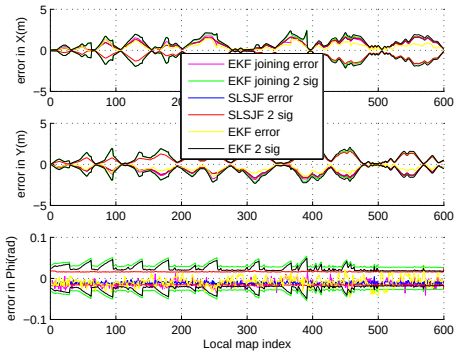
[5]All time measurements in this paper are performed on a laptop computer with Intel Core 2 Duo T7500 at 2.2GHz, 3GB of RAM and running Windows, with all programs written in MATLAB. More simulation results are available at the web site: http://services.eng.uts.edu.au/~sdhuang.



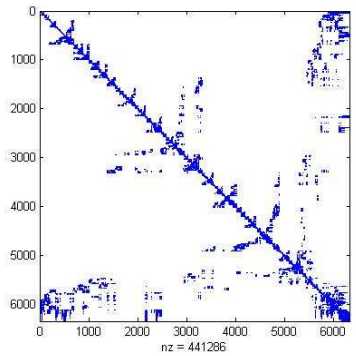
(a) The robot trajectory and the global map obtained by SLSJF (red:feature, green: robot poses) superimposed with the EKF SLAM map (black) and the global map by EKF map joining (blue)



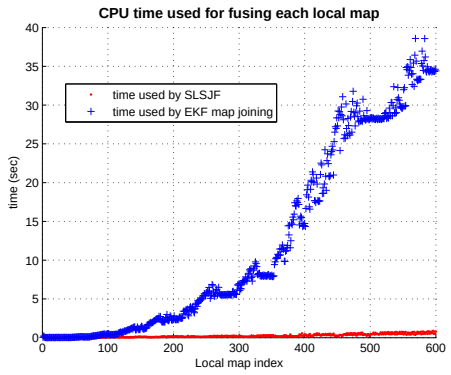
(b) A close look at the estimate of the five features at the upper-left corner of the map (a): dots (black) are true positions, solid ellipses (red) are from SLSJF, dotted ellipses (blue, coincide with red ones) are from EKF map joining, dashed ellipses (black) are from EKF SLAM



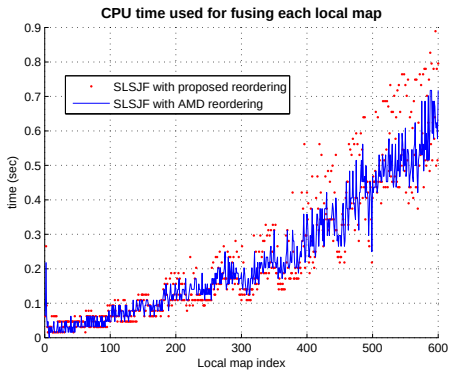
(c) Estimation errors of the 600 robot end poses by SLSJF, EKF map joining, and conventional EKF SLAM



(d) Sparse information matrix obtained by SLSJF (441286 non-zero elements in a 6340×6340 matrix)



(e) Time required for fusing each local map in SLSJF and EKF sequential map joining



(f) Comparison between the processing time using the proposed reordering and that using AMD reordering

Fig. 3. Simulation results

Algorithm	Data Association	Update	Others	Total
EKF map joining	12s	7287s	7s	7306s
SLSJF	87s	12s	46s	145s

TABLE I
PROCESSING TIME OF EKF SEQUENTIAL MAP JOINING AND SLSJF.

[3][4][13][14], presumably due to different parameters used by various researchers. The results in this section therefore only demonstrate that SLSJF can be applied to this popular data set.

Fig. 4(a) shows the map obtained by conventional EKF SLAM. The odometry and range-bearing observation data were used to build 200 local maps by EKF SLAM. Fig. 4(b) shows the global map obtained by joining the 200 local maps using SLSJF. Data association in SLSJF was performed using Nearest Neighbor method [8]. Fig. 4(c) shows all the non-zero elements of the information matrix in black. The information matrix is not very sparse because the sensor range is relatively large (around $80m$) as compared with the size of

the environment ($300m \times 300m$). Fig. 4(d) shows the CPU time required to fuse each of the 200 local maps. The total processing time for joining all the 200 maps by SLSJF is 22 seconds (the time required for building the 200 local maps is 63 seconds).

V. Related work and discussions

In this section, some of the properties of SLSJF and some related work are discussed.

A. Different ways to achieve sparse representation

The sparse representations of SLAM recently proposed in the literature (e.g. [1][2][3][4][15]) make use of different state vectors and/or have different strategies for marginalizing out robot poses. In SAM [2], incremental SAM (iSAM) [13], Tectonic SAM [11] and full-SLAM [5], all the robot poses are included in the state vector and no marginalization is needed. However, the dimension of the state vector is very high, especially when the robot trajectory is long.

When all the previous robot poses are marginalized out as in conventional EIF SLAM, the information matrix becomes dense although it is approximately sparse [16]. The Sparse Extended Information Filter (SEIF) presented in [1] approximates the information matrix by a sparse one using sparsification, but this leads to inconsistent estimates [3].

The Exactly Sparse Extended Information Filter (ESEIF) developed by [3] follows the conventional EIF SLAM algorithm, but marginalizes out the robot pose and relocates the robot from time to time. In this way the information matrix is kept exactly sparse by sacrificing the robot location information once in a while.

In Decoupled SLAM (D-SLAM) algorithm [4], the robot pose is not incorporated to the state vector for mapping. The observations made from one robot pose are first transferred into the relative position information among the observed features (the robot pose is marginalized out from the observations), then the relative position information is used to update the map. This process also results in some information loss.

The D-SLAM map joining algorithm [15] first builds local maps and then marginalizes out the robot start and end poses from the local map, the obtained relative position information among features are fused into the global map in a way similar to the D-SLAM algorithm. The odometry information is maintained in the local maps but there is still some information loss due to the marginalization of robot start/end poses.

In SLSJF, the robot start and end poses of the local maps are never marginalized but kept in the global state vector. Thus all the information from local maps is preserved.

If each local map is treated as one integrated observation, then SLSJF has some similarity to iSAM [13]. The role of local maps in SLSJF is also similar to the "star nodes" in the Graphical SLAM [17]. However, in the Graphical SLAM, the poses are first added in the graph and then "star nodes" are made. While in SLSJF, most of the robot poses are marginalized out during the local map building steps. Those robot poses are never present in the global state vector.

B. Computational complexity

The map joining problem considered in this paper is similar to that studied in [6] and [7]. The computational complexity of the local map building is $O(1)$ since the size of local map is small. The computational complexity of the global map update is $O(n^2)$ for the sequential map joining approach in [6] and the Constrained Local Submap Filter in [7].

In SLSJF, the robot start/end poses of the local maps are included in the global state vector and the EIF implementation results in an exactly sparse information matrix. This makes SLSJF much more efficient than the EKF sequential map joining [6][7].

Although simulation results show that SLSJF is computationally very efficient for large-scale SLAM, the computational complexity of several steps in SLSJF may not be $O(n)$ for worst case scenarios. For example, the number of fill-ins introduced in the Cholesky factorization depends on the environment and the robot trajectory. This influences the computational cost of the full Cholesky factorization step and the step of solving the sparse linear equations. Also, the computational cost of the proposed reordering is not well understood yet. In theory, SLSJF suffers the general $O(n^{1.5})$ cost for worst case scenario of planar grids, as all sparse factorization based methods do [18]. This is similar to the treemap algorithm [19] and the SAM using nested dissection algorithm [20].

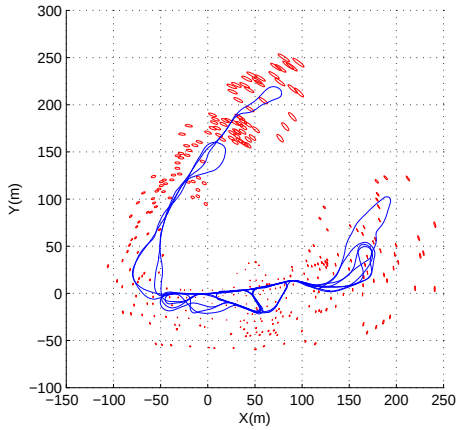
Very recently, it was shown in [21] that the total computational cost of local map building and map joining can be reduced to $O(n^2)$ by an EKF based "Divide and Conquer SLAM" (D&C SLAM). Although D&C SLAM was shown to be much more efficient than conventional EKF SLAM, it was not compared with the more efficient EKF sequential map joining [21].

The SLSJF has some similarity to the Tectonic SAM algorithm [11]. Tectonic SAM is also an efficient submap based approach and the state vector reordering and Cholesky factorization are used in solving the least-square problem. The submap fusion in Tectonic SAM uses a divide-and-conquer approach, which is more efficient than the sequential map joining in SLSJF when data association is assumed. The major difference between Tectonic SAM and SLSJF is that in Tectonic SAM, all the robot poses involved in building the local maps are kept and the dimension of the global state vector is much higher than that of SLSJF.

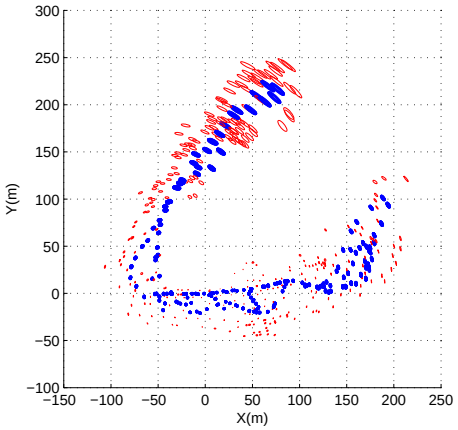
C. Requirements on SLSJF

In SLSJF, it is assumed that the local maps are consistent and accurate enough. If the local maps are inconsistent, SLSJF may produce wrong results due to the wrong information provided by the local maps. When the local maps are inaccurate, SLSJF may become inconsistent due to linearization errors.

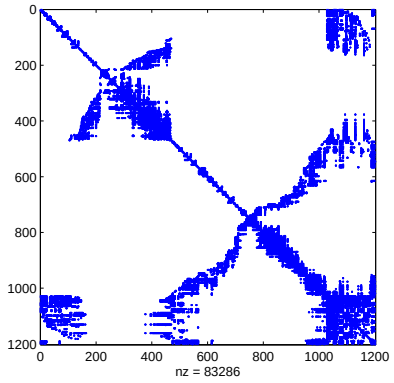
Another assumption made in SLSJF is that the local map only involves "nearby objects". This guarantees that the information matrix is exactly sparse no matter how many local maps are fused. When this assumption does not hold such as the case with vision sensors, SLSJF can still be applied since a significant number of feature pairs will not be present



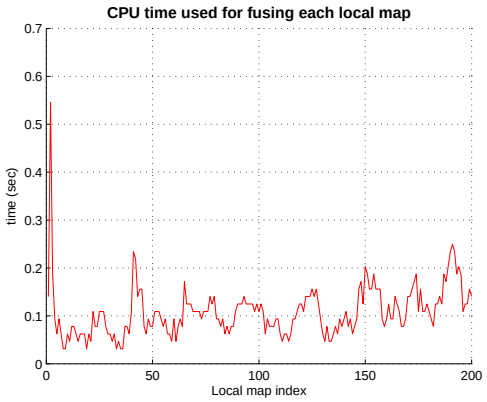
(a) EKF SLAM result



(b) Map obtained by joining 200 local maps using SLSJF



(c) Sparse information matrix obtained by SLSJF



(d) Processing time for fusing each of the 200 local maps

Fig. 4. The map joining results using Victoria Park data set.

concurrently in the same local map. However, the processes of selecting potentially matched features and reordering the state vector may need modifications to make the algorithm more efficient.

Similar to [6][7], there is no requirement on the structure of the environments for SLSJF to be applicable. This is different from the efficient treemap SLAM algorithm [19] where the environment has to be "topological suitable". Another difference between SLSJF and the treemap SLAM algorithm is that the covariance submatrix recovery and data association have been ignored in the treemap SLAM implementations available to date [19][22][23].

D. Exact covariance submatrix recovery

The covariance submatrix recovery in SLSJF is exact. This is different from the approximate covariance submatrix recovery methods (e.g. [1] [10]) where only an approximate or upper bound of covariance submatrix is computed. As pointed out in [10], the upper bound can only be used in nearest neighbor data association [8] but cannot be used in the more robust joint compatibility test [12].

An algorithm for exact recovery of covariance submatrix was proposed in iSAM [13]. It has "$O(n)$ time complexity for band-diagonal matrices and matrices with only a constant number of entries far from the diagonal, but can be more expensive for general sparse matrices" [13]. The covariance submatrix recovery in SLSJF is similar. The major advantages of SLSJF over iSAM is that the dimension of the state vector in SLSJF is much lower than that of iSAM. Thus SLSJF may be more suitable for the situations where the robot trajectory is very long and/or the observation frequency is high, which is true for many common sensors such as laser range finders.

E. Incremental Cholesky factorization for recovery

The idea of incrementally computing the Cholesky factorization is motivated by [4]. The main difference between the recovery method in SLSJF and that in [4] is that complete Cholesky factorization and direct method for linear equation solving are used in SLSJF, while approximate Cholesky factorization and Preconditioned Conjugate Gradient method were used in [4].

The incremental Cholesky factorization also has some similarity with the QR factorization update in [13]. The QR factorization update in [13] is based on "Givens rotations", while the incremental Cholesky factorization process in SLSJF is based on the "block-partitioned form of Cholesky factorization". The performance of these two approaches are expected to be similar.

F. Reordering of the global state vector

In SLSJF, the reordering of state vector aims to combine the advantages of AMD reordering (where the number of fill-ins is reduced [2][13]) and the reordering by distance (where the efficient incremental Cholesky factorization procedure can be applied in most cases [4]).

The idea behind the "reordering by distance" is to make sure that the robot observes only the features that are in the bottom part of the state vector for as long as possible no matter in which direction the robot is moving. However, this is not the best way of reordering for indoor environments where features in different rooms might actually be very close but cannot be seen simultaneously. For indoor environments, the knowledge on the structure of the environment (and the knowledge on the possible robot trajectory) can be exploited to place "the features that are likely to be re-observed" near the bottom of the state vector.

G. Consistency

The SLSJF algorithm does not contain any approximations (such as sparsification [1]) that can lead to estimator inconsistency. However, as the case with all EKF/EIF based estimation algorithms, it is possible that inconsistencies occur in SLSJF due to errors introduced by the linearization process.

It has been suggested that local map based strategies can improve the consistency of SLAM by keeping the robot orientation error small [24][25]. We had conducted many simulations and found that this is true for some scenarios especially when the process noise, the feature density and the sensor range are all small, or sequential update is used in EKF when multiple features can be observed from one robot pose. In many practical scenarios, for example, in the simulation results presented in Section IV-A, we found that both EKF SLAM (with batch update) and map joining results are consistent, mainly due to the small observation and odometry noises and the high feature density. When noise values were gradually increased both strategies became inconsistent, almost always at the same level of noise. This is likely due to the fact that in any submap joining algorithm, inconsistency in even one of the submaps, leads to an inconsistent global map.

In SLSJF, all the robot start/end poses are in the global state vector and there is no prediction step within the EIF. Thus the SLSJF can be treated as a linearized least square solution with only one iteration in each map fusion step. In fact, at any map fusion step, the linearization error can be reduced further by recomputing the information matrix I and the information vector i as a sum of all the contributions in (8) using the new estimate as linearization point for the Jacobians. This process is able to improve the consistency significantly, but with more computational cost.

H. Treating the local map as a virtual observation

Many submap based SLAM algorithms (either explicitly or implicitly) treat the local map as a virtual observation, but most of them treat a local map as "an observation made from the robot start pose to all the features in the local map". In SLSJF, the local map is treated as "an observation made from the robot start pose to all the features in the local map and a virtual robot located at the robot end pose". This motivates the inclusion of all the robot start/end poses in the global state vector to achieve exactly sparse information matrix.

I. Comparison with two level mapping algorithms

The output of SLSJF is one single global stochastic map. This approach is different from the two level mapping algorithms (e.g. Hierarchical SLAM [26], Atlas [27], Network Coupled Feature Maps [28]), where a set of local maps are maintained and the relationship among these maps is described at a higher level. Though promising due to their reduced computational cost, the two level mapping approaches require more work to completely resolve the question of how to treat the overlapping regions among local maps. As pointed out in [26], all the two level mapping systems result in suboptimal solutions because the effect of the upper level update cannot be propagated back to the local level.

VI. Conclusions

By adding robot start/end poses of the local maps into the global state vector, an exactly sparse extended information filter for local submap joining, SLSJF, is developed. There is no approximation involved in SLSJF apart from linearization processes. SLSJF contains not only the filter steps but also two important steps that are essential for real world application of EIF based algorithms — a covariance submatrix recovery step and a data association step. The sparse information matrix together with the novel state vector and covariance submatrix recovery procedure make the SLSJF algorithm computationally very efficient.

SLSJF achieves an exactly sparse information matrix with no information loss. The dimension of its state vector is significantly less than that of the full SLAM algorithm [5] where all the robot poses are included in the state vector. As it does not matter how the local maps are built, SLSJF can also be applied to large-scale range-only or bearing-only SLAM problem — first use range-only or bearing-only SLAM algorithms to build local maps and then fuse the local maps together using SLSJF.

For the successful application of SLSJF for local map joining, it is important that all the local maps are consistent. Thus it is essential to use reliable SLAM algorithms to build the local maps.

More work is required to determine the best reordering strategy for SLSJF, to improve the robustness of SLSJF to linearization errors, and to extend SLSJF to 3D local map joining. Research along these directions is underway.

Acknowledgment

The authors would like to thank Dr. Udo Frese for very helpful suggestions and the anonymous reviewers for the valuable comments.

REFERENCES

[1] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, pp. 693-716, 2004.

[2] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing". *International Journal of Robotics Research*, vol. 25, no. 12, December 2006, pp. 1181-1203.

[3] M. R. Walter, R. M. Eustice and J. J. Leonard, "Exactly sparse Extended Information Filters for feature-based SLAM". *International Journal of Robotics Research*, vol. 26, no. 4, 2007, pp. 335-359.

[4] Z. Wang, S. Huang and G. Dissanayake, "D-SLAM: A decoupled solution to simultaneous localization and mapping", *International Journal of Robotics Research*, vol. 26, no. 2, February 2007, pp. 187-204.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.

[6] J. D. Tardos, J. Neira, P. M. Newman and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data", *International Journal of Robotics Research*, vol. 21, no. 4, April 2002, pp. 311-330.

[7] S. B. Williams, *Efficient Solutions to Autonomous Mapping and Navigation Problems*, PhD thesis, Australian Centre of Field Robotics, University of Sydney, 2001. available online http://www.acfr.usyd.edu.au/

[8] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. on Robotics and Automation*, vol. 17, pp. 229–241, 2001.

[9] T. P. Speed, and H. T. Kiiveri, "Gaussian Markov distributions over finite graphs". *The Annals of Statistics*, 14(1): 138-150, 1986.

[10] R. M. Eustice, H. Singh, J. J. Leonard and M. R. Walter, "Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters", *International Journal of Robotics Research*. 25(12): 1223-1242, 2006.

[11] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact, out-of-core, submap-based SLAM," *In Proceedings of 2007 IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 10-14 April 2007, pp. 1678-1685.

[12] J. Neira and J.D. Tardos, "Data association in stochastic mapping using the joint compatibility test", *IEEE Trans. Robotics and Automation*, vol. 17, no. 6, pp. 890-897, Dec 2001.

[13] M. Kaess, A. Ranganathan, F. Dellaert, "iSAM: Fast incremental Smoothing and Mapping with efficient data association," *In Proceedings of 2007 IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 10-14 April 2007, pp. 1670-1677.

[14] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map building (SLAM) algorithm for real time implementation," *IEEE Trans. on Robotics and Automation*, vol. 17, pp. 242-257, 2001.

[15] S. Huang, Z. Wang, and G. Dissanayake. "Mapping large-scale environments using relative position information among landmarks". *In Proceedings of 2006 International Conference on Robotics and Automation*, pp. 2297-2302, 2006.

[16] U. Frese, "A proof for the approximate sparsity of SLAM information matrices". *In Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pp.331-337. 2005.

[17] J. Folkesson and H.I. Christensen, "Closing the loop with Graphical SLAM", *IEEE Transactions on Robotics*, 2007, vol. 23, no. 4, pp. 731-741.

[18] R.J. Lipton and D.J. Rose and R.E. Tarjan, "Generalized nested dissection", *SIAM Journal on Numerical Analysis*, 1979, vol. 16, no. 2, pp. 346-358.

[19] U. Frese, "Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping". *Autonomous Robots*, 21(2): 103-122, 2006.

[20] P. Krauthausen, F. Dellaert, and A. Kipp, "Exploiting locality by nested dissection for square root smoothing and mapping", *In Proceeding of Robotics: Science and Systems*, Philadelphia, U.S.A. 2006.

[21] L. M. Paz, J. Guivant, J. D. Tardos, and J. Neira, "Data association in O(n) for Divide and Conquer SLAM," *In Proceedings of 2007 IEEE International Conference on 2007 Robotics: Science and Systems*, June 27-30, Atlanta, USA

[22] U. Frese ans L. Schroder, "Closing a million-landmarks loop". *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 9 - 15, 2006, pp. 5032-5039.

[23] U. Frese, "Efficient 6-DOF SLAM with Treemap as a generic backend," *In Proceedings of 2007 IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 10-14 April 2007, pp. 4814-4819.

[24] J.A. Castellanos, R. Martinez-Cantin, J.D. Tardos and J. Neira. "Robocentric map joining: Improving the consistency of EKF-SLAM". *Robotics and Autonomous Systems*, 2007, vol. 55, 21-29.

[25] S. Huang and G. Dissanayake, "Convergence and consistency analysis for Extended Kalman Filter based SLAM". *IEEE Transactions on Robotics*, 2007, vol. 23, no. 5, 1036-1049.

[26] C. Estrada, J. Neira and J.D. Tardos. "Hierarchical SLAM: real-time accurate mapping of large environments". *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588-596, 2005.

[27] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller, "SLAM in large-scale cyclic environments using the atlas framework", *International Journal of Robotics Research*. 23(12), pp. 1113-1139, 2004.

[28] T. Bailey, *Mobile Robot Localization and Mapping in Extensive Outdoor Environment*. Ph.D. Thesis. Australian Centre of Field Robotics, University of Sydney. 2002. available online http://www.acfr.usyd.edu.au/

Shoudong Huang was born on December 8, 1969. He received the Bachelor and Master degrees in Mathematics, Ph.D in Automatic Control from Northeastern University, P.R. China in 1987, 1990, and 1998, respectively. He is currently a Senior Lecturer at Australian Research Council (ARC) Centre of Excellence for Autonomous Systems, Faculty of Engineering, University of Technology, Sydney, Australia. His research interests include nonlinear control systems and mobile robots mapping, exploration and navigation.

Zhan Wang received the B.E. degree from the Harbin Institute of Technology, P.R. China, in 1998, and the Ph.D. degree in engineering from the University of Technology, Sydney, Australia, in 2007. He is currently working as a postdoctoral research fellow at the ARC Centre of Excellence for Autonomous Systems, Faculty of Engineering, University of Technology, Sydney, Australia. His research interests include SLAM for mobile robots, estimation theory and computer vision.

Gamini Dissanayake is the James N Kirby Professor of Mechanical and Mechatronic Engineering at University of Technology, Sydney (UTS). His current research interests are in the areas of localization and map building for mobile robots, navigation systems, dynamics and control of mechanical systems, cargo handling, optimization and path planning. He leads the UTS node of the ARC Centre of Excellence for Autonomous Systems. He graduated in Mechanical/Production Engineering from the University of Peradeniya, Sri Lanka. He received his M.Sc. in Machine Tool Technology and Ph.D. in Mechanical Engineering (Robotics) from the University of Birmingham, England in 1981 and 1985, respectively.