2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Exact State and Covariance Sub-matrix Recovery for Submap Based Sparse EIF SLAM Algorithm

Shoudong Huang, Zhan Wang and Gamini Dissanayake

Abstract— **This paper provides a novel state vector and covariance sub-matrix recovery algorithm for a recently developed submap based exactly sparse Extended Information Filter (EIF) SLAM algorithm — Sparse Local Submap Joining Filter (SLSJF). The algorithm achieves exact recovery instead of approximate recovery. The recovery algorithm is very efficient because of an incremental Cholesky factorization approach and a natural reordering of the global state vector. Simulation results show that the computation cost of the SLSJF is much lower as compared with the sequential map joining algorithm using Extended Kalman Filter (EKF). The SLSJF with the proposed recovery algorithm is also successfully applied to the Victoria Park data set.**

I. INTRODUCTION

In the recent years, sparse representations for solving Simultaneous Localization and Mapping (SLAM) problems have attracted special attention since the development of the Sparse Extended Information Filter (SEIF) by Thrun et al. [1]. It has been shown that significant computational advantages can be achieved by exploiting the sparseness of the information matrix or techniques from sparse graph and sparse linear algebra ([2]-[5]).

Local submap joining [6][7] provides an efficient way to build large-scale maps. The idea of local submap joining is to first build a sequence of small sized local submaps (e.g. by traditional Extended Kalman Filter (EKF) SLAM [8]), and then combine the local submaps into a large-scale global map. A key advantage of local submap joining is that the computation cost of submap building is constant and the frequency of the computationally expensive global map update is significantly reduced [7].

Combining the local submap joining idea with sparse representation can make SLAM algorithms even more efficient. A number of such algorithms, for example, the tree-map algorithm [9], Tectonic SAM [10], and D-SLAM submap joining [11], are available. However, in the tree-map algorithm [9] and Tectonic SAM [10], data association is assumed and thus the recovery of covariance sub-matrix is not needed. In the algorithm presented in D-SLAM submap joining [11], the robot pose information in each submap is not exploited and this results in some information loss.

It is well known that the recovery of state vector and covariance sub-matrix is one of the most computationally intensive steps in the EIF SLAM implementation. In [1] and [12], efficient approaches for approximately recovering the covariance sub-matrix are given but the inaccurate covariance may result in wrong data association [12].

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

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

This paper studies the exact state vector and covariance sub-matrix recovery for a recently developed submap based sparse EIF SLAM algorithm – Sparse Local Submap Joining Filter (SLSJF) [13]. The major components of the recovery algorithm include a natural reordering of the global state vector and an incremental Cholesky factorization method. Simulation and experimental results show the effectiveness of the proposed recovery strategy.

The paper is organized as follows. Section II briefly describes the SLSJF algorithm. The state and covariance sub-matrix recovery algorithm is stated in details in Section III. Section IV provides simulation and experimental results. Some related work are discussed in Section V and Section VI concludes the paper.

II. SPARSE LOCAL SUBMAP JOINING FILTER

This section briefly describes the Sparse Local Submap Joining Filter (SLSJF) [13] algorithm.

A. Overall structure of submap joining

The objective of local submap joining is to combine the local submaps and obtain a global map containing all the features. Similar to the sequential submap joining in [6], SLSJF fuses the local submaps one by one. That is, first set local submap 1 as the global map, then fuse local map k ($k \geq 2$) into the global map in sequence.

B. Local submap

It is assumed that a consistent local submap can be constructed by some SLAM algorithm and is expressed by

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

where $\hat{X}^L$ (here the superscript 'L' stands for the "local" submap) 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 final robot pose X_r^L and all the local feature positions $X_1^L, \cdots, X_n^L$ (as in traditional EKF SLAM). The coordinate system of a local submap is defined by the initial robot pose when the building of the local submap is started, i.e. the robot starts at the coordinate origin of the local submap.

C. Relationship between consecutive submaps

In order to fuse the local maps together into a global map, some relationship between the submaps must be known.

In the SLSJF stated below, it is assumed that the robot starts to build local submap $k+1$ as soon as it finishes local submap k. So the relationship between submaps is: the robot end pose in local submap k and the robot start pose in local submap $k+1$ are identical (Fig. 1).

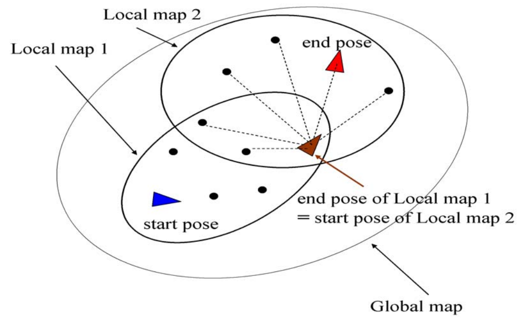


Fig. 1. The relationship between submaps in SLSJF

In the case when the robot moves to a new pose after it finishes local submap k and then starts to build local submap $k+1$, the relationship between the two local submaps is described by the robot motion from the robot end pose in local submap k to the robot start pose in local submap $k+1$. This general case is treated in [13].

D. Global map state vector

In SLSJF, the global map state vector contains the position of all the features and the robot end poses of each local submaps.

The global map state vector in SLSJF is

$$X^G = (X_1^G, \cdots, X_{n_1}^G, X_{1e}^G, X_{n_1+1}^G, \cdots, X_{n_1+n_2}^G, X_{2e}^G, \cdots) \tag{3}$$

where $X_1^G, \cdots, X_{n_1}^G$ are the global position of the features in local submap 1, $X_{n_1+1}^G, \cdots, X_{n_1+n_2}^G$ are the global position of those features in local submap 2 but not in local submap 1. $X_{ke}^G = (x_{ke}^G, y_{ke}^G, \phi_{ke}^G)$ is the global position of the robot end pose in local submap k ($1 \le k \le p$ where p is the total number of local submaps). Here the supscript 'G' stands for the global map; the subscript 'e' stands for 'robot end pose'. Note that the robot end pose of local map k is the same as the robot start pose of local map $k+1$.

E. Treat submap as an observation

Since a local submap is a consistent estimate of the relative position from robot start pose to the local features and the robot end pose (Fig. 1), it can be treated as an observation — a function of the global state vector (with noise). Suppose local submap $k+1$ is given by (1) and the data association result is $X_1^L \leftrightarrow X_{i1}^G, \cdots, X_n^L \leftrightarrow X_{in}^G$. Then the local map can be regarded as an observation

$$z_{map} = \hat{X}^L = H_{map}(X^G) + w_{map} \tag{4}$$

where $H_{map}(X^G)$ is given by

$$\begin{pmatrix} (x_{(k+1)e}^G - x_{ke}^G)\cos\phi_{ke}^G + (y_{(k+1)e}^G - y_{ke}^G)\sin\phi_{ke}^G \\ (y_{(k+1)e}^G - y_{ke}^G)\cos\phi_{ke}^G - (x_{(k+1)e}^G - x_{ke}^G)\sin\phi_{ke}^G \\ \phi_{(k+1)e}^G - \phi_{ke}^G \\ (x_{i1}^G - x_{ke}^G)\cos\phi_{ke}^G + (y_{i1}^G - y_{ke}^G)\sin\phi_{ke}^G \\ (y_{i1}^G - y_{ke}^G)\cos\phi_{ke}^G - (x_{i1}^G - x_{ke}^G)\sin\phi_{ke}^G \\ \vdots \\ (x_{in}^G - x_{ke}^G)\cos\phi_{ke}^G + (y_{in}^G - y_{ke}^G)\sin\phi_{ke}^G \\ (y_{in}^G - y_{ke}^G)\cos\phi_{ke}^G - (x_{in}^G - x_{ke}^G)\sin\phi_{ke}^G \end{pmatrix}$$

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

F. Global map update

In the SLSJF algorithm, information vector and information matrix are used to express the Gaussian distribution of the global state estimate. Let $i(k)$ represent the information vector and $I(k)$ be the associated information matrix of the global map before the fusion of local submap $k+1$. The relationship among the global state vector estimate $\hat{X}^G(k)$, the corresponding covariance matrix $P(k)$, the information vector, and the information matrix is ([5])

$$i(k) = P(k)^{-1}\hat{X}^G(k), \quad I(k) = P(k)^{-1}. \tag{5}$$

When the local map $k+1$ is to be fused, the initial value of the global position of new features and robot end pose are first computed and the corresponding state vector are enlarged. The dimension of $i(k)$ and $I(k)$ are increased by simply adding zero elements corresponding to the new features and robot pose; this results in new expressions for $\hat{X}^G(k), i(k)$ and $I(k)$. Then the observation z_{map} is used to update the information vector and the information matrix as follows:

$$\begin{array}{rcl} I(k+1) & = & I(k) + \nabla H_{map}^T R_{map}^{-1} \nabla H_{map} \\ i(k+1) & = & i(k) + \nabla H_{map}^T R_{map}^{-1} [z_{map}(k+1) \\ & & -H_{map}(\hat{X}^G(k)) + \nabla H_{map}\hat{X}^G(k)] \end{array} \tag{6}$$

where ∇H_{map} is the Jacobian of the function H_{map} with respect to all the states evaluated on the current state estimate $\hat{X}^G(k)$.

G. Exactly sparse information matrix

It is well known that the direct implementation of EIF in SLAM result in dense (although approximately sparse) information matrix [1] due to the marginalization of previous robot poses [3]. The main reason why the information matrix $I(k)$ in SLSJF is exactly sparse is that there is no marginalization involved in the global state. The only marginalization performed is done within the local maps where most of the robot poses are marginalized out.

III. THE STATE VECTOR AND COVARIANCE SUB-MATRIX RECOVERY ALGORITHM

Although the global map update can be performed efficiently in the information form (6), the state vector is needed to compute $i(k+1)$ and to evaluate the Jacobians. Before fusing submap $k+2$ to the global map containing submaps 1 to $k+1$, data association is necessary to find the features in local submap $k+2$ that are already included in the global map and their corresponding indices in the global state vector.

To perform the data association using Nearest Neighbor [8] or Joint Compatibility Test [14], the covariance sub-matrix corresponding to the potentially matched features is needed.

A. Reorder the global map state vector when necessary

After the update step described in (6), the global state vector is reordered if the following condition holds. The purpose of reordering the state vector is to make the state vector recovery and the covariance sub-matrix recovery process more efficient. Especially, the reordering aims to (i) reduce the number of fill-ins in the Cholesky factorization, and, (ii) make the efficient incremental Cholesky factorization (Section III-B) more applicable.

Reordering condition: One or more of the features in local submap $k+1$ is not located within the bottom n_0 elements in the global state vector. [1]

Reordering strategy: When the global state vector is to be reordered, the features and robot poses in the global state vector are divided into two parts. The final robot pose $X^G_{(k+1)e}$ and the features that are within distance d_0 [2] from the robot pose $\hat{X}^G_{(k+1)e}$ are placed in the bottom part of the state vector and they are ordered by their distances to $\hat{X}^G_{(k+1)e}$. The smaller the distance, the closer to the bottom. All the other features and robot poses are placed in the upper part of the state vector and they are reordered by Approximate Minimal Degree (AMD) algorithm [2].

In general, whenever the robot closes a big loop, the state vector will be reordered. Once the state vector is reordered, the corresponding information matrix $I(k+1)$ and information vector $i(k+1)$ are reordered accordingly. They are still denoted as $I(k+1)$ and $i(k+1)$.

B. Incremental Cholesky factorization

By (5), the state vector estimate $\hat{X}^G(k+1)$ can be recovered by solving the sparse linear equation

$$I(k+1)\hat{X}^G(k+1) = i(k+1). \tag{7}$$

Any column of the covariance matrix can also be obtained by solving a similar sparse linear equation [12]. Similar to SAM [2], direct linear equation solver using Cholesky factorization is applied to solve these sparse linear equations.

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

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

[2]The threshold d_0 is related to n_0. It also depends on the feature density of the environment and the guideline is that the dimension of the features and robot poses that are placed in the bottom part of the state vector is around n_0.

Case (i). If the global state vector was not reordered in Section III-A, then the two sparse information matrices, $I(k+1)$ and $I(k)$, are very similar. In this case, the Cholesky factorization of $I(k)$ is used to construct the Cholesky factorization of $I(k+1)$.

In fact, by (6),

$$I(k+1) = I(k) + \begin{bmatrix} 0 & 0 \\ 0 & \Omega \end{bmatrix} \tag{8}$$

where the upper-left element in Ω is non-zero. Here Ω is determined by the term $\nabla H^T_{map}(R_{map})^{-1}\nabla H_{map}$ in (6). Its dimension is less than n_0 since otherwise the state vector would have been reordered.

Suppose the Cholesky factorization of $I(k)$ is L_k (a lower triangular matrix). Let L_k and $I(k)$ be partitioned according to (8) as

$$L_k = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}, \quad I(k) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22} \end{bmatrix}. \tag{9}$$

According to (8) and (9), $I(k+1)$ can be expressed by

$$I(k+1) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & \tilde{I}_{22} \end{bmatrix} = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}+\Omega \end{bmatrix}. \tag{10}$$

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

$$L_{k+1} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & \tilde{L}_{22} \end{bmatrix} \tag{11}$$

where $\tilde{L}_{22}$ is the Cholesky factorization of a low dimensional matrix $\Omega + L_{22}L_{22}^T = \tilde{I}_{22} - L_{21}L_{21}^T$.

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

Now we can see that the reordering technique described in Section III-A has a number of advantages:

- The reordering makes the incremental Cholesky factorization described in Case (i) more efficient, since the dimension of matrix $\Omega + L_{22}L_{22}^T$ will be less than n_0.
- Once a reorder is performed, another reordering will not happen in the next few steps. This is because the robot can not move very far away from local submap $k+1$ in the next few steps. No matter which direction it moves along, it can not observe old features that are not located in the bottom part of the state vector until it travels more than distance d_0. Thus Case (i) can be applied in most of the cases.
- Since Approximate Minimal Degree (AMD) algorithm is applied to the upper part of the state vector, the reordering will significantly reduce fill-ins in the Cholesky factorization [2]. This makes the linear equation solving more efficient.

C. Recovery of state vector and covariance sub-matrix

Before fusing submap $k+2$, the state vector estimate $\hat{X}^G(k+1)$ and the covariance sub-matrix need to be recovered from $I(k+1)$ and $i(k+1)$.

1) Recover the state vector: Once the Cholesky factorization L_{k+1} is obtained in Section III-B, the sparse linear equation (7) can be solved easily by solving $L_{k+1}Y = i(k+1)$ and $L_{k+1}^T \hat{X}^G(k+1) = Y$ because $L_{k+1}L_{k+1}^T = I(k+1)$.

2) Find the potentially matched features: Since the size of local map $k+2$ is small, it is not necessary to compare the features in local map $k+2$ with the global map features which are far away from the "origin" of local submap $k+2$.

The potentially overlapping local maps are selected by computing the distances from each local submap origin to the origin of local submap $k+2$. The estimate of the origins are available from the global state vector. Moreover, for each feature in the potentially overlapping local maps, if its distance to $\hat{X}^G_{(k+1)e}$ is larger than the radius of local map $k+2$ plus the maximal possible estimation error, it can not be matched with any feature in local map $k+2$.

Furthermore, the (estimated) relative positions from the potentially matched features to $\hat{X}^G_{(k+1)e}$ are computed. They are compared with each feature in local map $k+2$. For each potentially matched feature, if its Euclidean distance to the closest feature in local map $k+2$ is larger than a threshold, then it cannot match with any local map feature and will be removed from the list of potentially matched features.

3) Recover the covariance sub-matrix: The covariance sub-matrix corresponding to the potentially matched features can be obtained by first computing the corresponding columns of the whole covariance matrix and then deleting the unnecessary rows. Each column of the covariance matrix can be obtained similarly as the state vector.

IV. SIMULATION AND EXPERIMENT RESULTS

In this section, simulation and experiment results are given to illustrate the accuracy and efficiency of the SLSJF with the proposed recovery algorithm applied.

A. Simulation results

The $150 \times 150m^2$ simulated environment contains 2500 features arranged in uniformly spaced rows and columns. The robot starts from the left bottom corner of the square and follows a random trajectory (Fig. 2(a)), revisiting many features and closing many loops. A sensor with a field of view of 180 degrees and a range of 6 meters is simulated to generate relative range and bearing measurements between the robot and the features. In total, there are 22384 loops (22384 robot poses) and 138026 observations are made from the robot to 2318 features.

Five hundred small sized local submaps were built by EKF SLAM. Each local submap contains around 10 features. SLSJF is implemented with the proposed recovery strategy applied. Data association is performed using Nearest Neighbor [8]. Fig. 2(b) shows the global map generated by fusing all the 500 local submaps using SLSJF. The robot trajectory can be seen approximately from the 500 robot poses. Fig. 2(a) shows the global map generated by fusing all the 500 local submaps using EKF sequential map joining [6][7].

Close examination shows that the global feature position estimate using SLSJF and EKF sequential map joining are both consistent since the feature positions fall within the 2σ covariance ellipses drawn around the estimates. Also, the map uncertainty of the two global maps are very similar. Fig. 2(d) shows the errors and 2σ bounds of the estimates of robot end poses when fusing each local submap. Comparing it with the robot end poses estimate from EKF sequential map joining shown in Fig. 2(c), it can be seen that the estimate from the two different approach are almost identical.

Figure 2(e) shows the non-zero elements of the sparse information matrix in black. Fig. 2(f) shows the CPU time [1] required for the local submap fusion time using SLSJF and EKF sequential map joining.

The total time for fusing all the 500 local submaps is 129 seconds for SLSJF and 7240 seconds for EKF sequential map joining (building the 500 local submaps takes 409 seconds, it takes traditional EKF SLAM more than 21 hours to finish the map).

B. Experimental results

The SLSJF was also applied to the popular Victoria Park data set which was first used in [15]. Neither ground truth nor sensor noise parameters are available for this data set. Published results for the vehicle trajectory and uncertainty estimates vary [3][4][15][18], 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.

Two hundred local submaps are built by EKF SLAM using the data set. Fig. 3(a) shows the global map obtained by joining 200 local submaps using SLSJF with the proposed state and covariance sub-matrix recovery algorithm. Data association is performed using Nearest Neighbor method [8]. Fig. 3(b) shows the CPU time required to fuse each local maps. The total computation time for joining all the 200 submaps by SLSJF is 22 seconds.

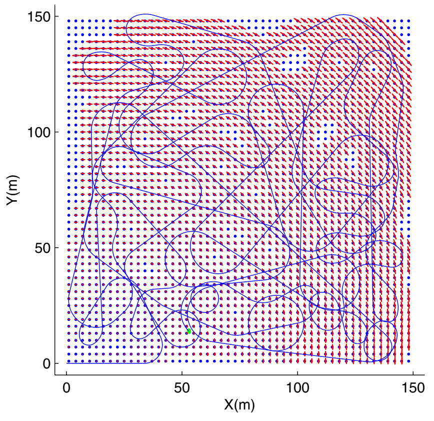
V. RELATED WORK AND DISCUSSIONS

Very recently, it is shown in [16] that the average computational cost of map joining in each step can be reduced to $O(n)$ by "Divide and Conquer SLAM". However, the algorithm is only compared to EKF SLAM instead of the more efficient EKF sequential map joining. Moreover, linear computation cost for data association is impossible unless some approximation is applied [16].

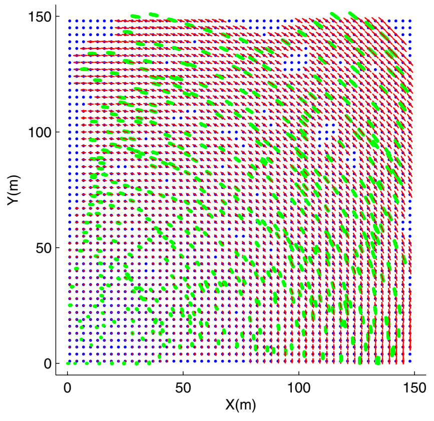
When the environment has an efficient tree representation and the hierarchical tree partitioning subalgorithm can find such a representation, the treemap SLAM algorithm can be very fast [19]. However, the covariance sub-matrix recovery and data association was ignored in the treemap SLAM implementations [9] [19].

The covariance sub-matrix recovery algorithms proposed in this paper is exact. This is different from the approximate covariance sub-matrix recovery methods (e.g. [1] [12]) where

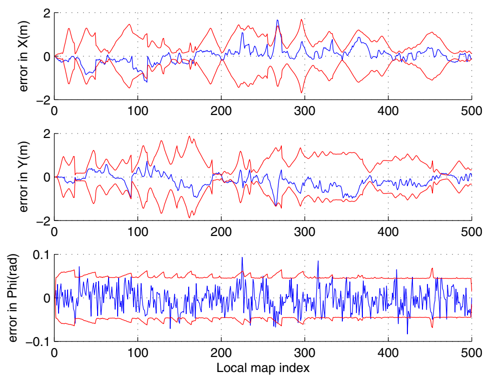
[1]All time measurements in this paper are performed on a laptop computer with Intel Core 2 Duo T7500 at 2.2GHz, 3GB of RAM and running Windows, all programs are written in MATLAB.



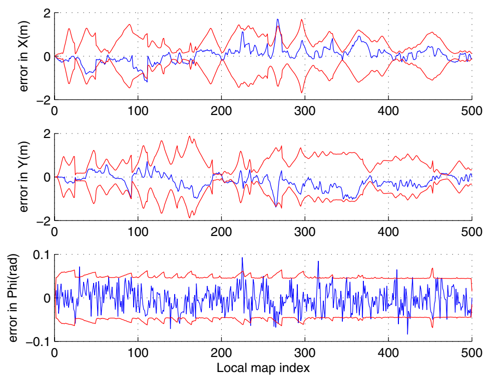
(a) The robot trajectory and the global map obtained by EKF sequential map joining



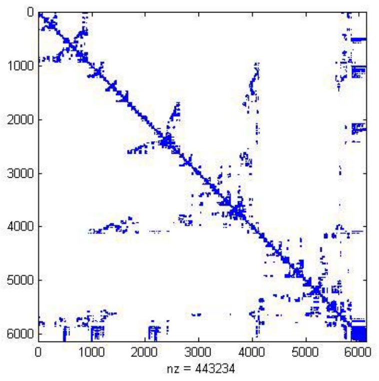
(b) Global map obtained by SLSJF



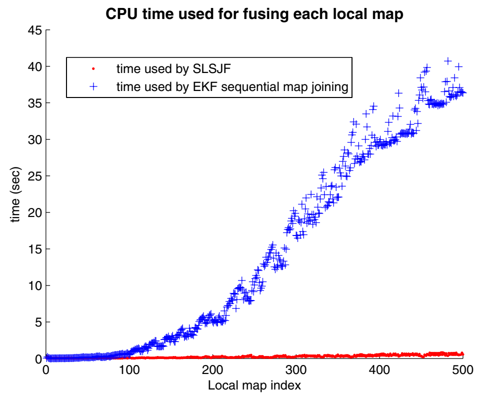
(c) Estimation error of local submap robot end poses by EKF sequential map joining



(d) Estimation error of local submap robot end poses by SLSJF



(e) Sparse information matrix obtained by SLSJF (443234 non-zero elements in a 6136 × 6136 matrix)

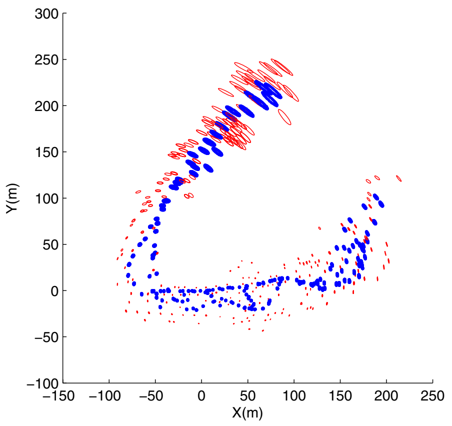


(f) Time used for fusing each local submap in SLSJF and EKF sequential map joining

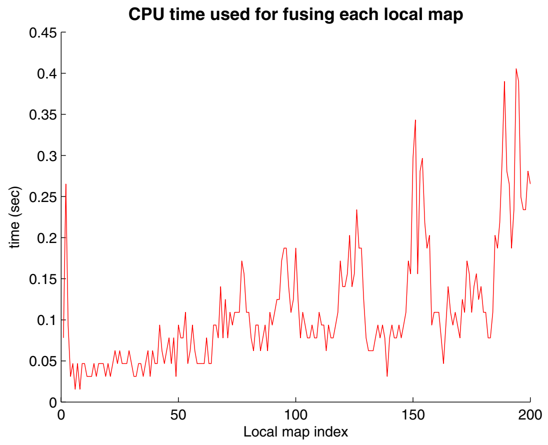
Fig. 2. Simulation results by EKF sequential map joining and SLSJF

only an approximate or upper bound of covariance sub-matrix is computed. The proposed recovery algorithm is similar to the exact recovery of covariance sub-matrix proposed in [18]. The idea of using incremental Cholesky factorization is motivated by [4]. The incremental Cholesky factorization has some similarity with the QR factorization update in [18]. However, the QR factorization update in [18] is based on "Givens rotations", while the proposed incremental Cholesky factorization process is based on the "block-partitioned form of Cholesky factorization". The major advantages of SLSJF over iSAM is that the dimension of the state vector in SLSJF is much lower than that in [18].

The reordering of global state vector is motivated by [4] and [2][18][17] while the reordering of state vector in [4] is



(a) Global map by joining 200 local submaps using SLSJF



(b) Time used for fusing each of the 200 local submaps

Fig. 3. The SLSJF map joining results using Victoria Park data set.

based on distances only and the reordering in [2][18][17] is based on AMD or Nested Dissection (ND) only. In SLSJF, the reordering of state vector aims to combine the advantages of AMD reordering (the number of fill-ins is reduced) and the advantages of reordering by distance (the efficient incremental Cholesky factorization procedure can be applied in most cases).

VI. CONCLUSIONS

This paper provides a novel method to exactly recover the state vector and covariance sub-matrix for Sparse Local Submap Joining Filter (SLSJF). There are two critical steps that make the recovery very efficient. One is the occasionally reordering of the global state vector. The other is an incremental Cholesky factorization method. The recovery procedure is applied in SLSJF and simulation results show that the computational complexity of the SLSJF algorithm is much less than that of sequential submap joining using EKF.

The SLSJF algorithm can also be applied to bearing-only or range-only large scale mapping. However, to build consistent local maps using bearing-only or range only sensors is still not a completely solved problem. The extension of the SLSJF to 3D SLAM is also an interesting future work.

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

[10] 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.

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

[12] 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.

[13] Z. Wang, *Exactly Sparse Extended Information Filters for SLAM*. Ph.D. Thesis. University of Technology, Sydney. 2007. available online http://services.eng.uts.edu.au/˜sdhuang

[14] 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.

[15] 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.

[16] 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

[17] 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.

[18] 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.

[19] 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.