Noname manuscript No.
(will be inserted by the editor)

Iterated D-SLAM Map Joining

Evaluating its performance in terms of consistency, accuracy and efficiency

Shoudong Huang · Zhan Wang · Gamini Dissanayake · Udo Frese

Received: date / Accepted: date

Abstract This paper presents a new map joining algorithm and a set of metrics for evaluating the performance of mapping techniques.

The input to the new map joining algorithm is a sequence of local maps containing the feature positions and the final robot pose in a local frame of reference. The output is a global map containing the global positions of all the features but without any robot poses. The algorithm builds on the D-SLAM mapping algorithm [1] and uses iterations to improve the estimates in the map joining step. So it is called Iterated D-SLAM Map Joining (I-DMJ). When joining maps I-DMJ ignores the odometry information connecting successive maps. This is the key to I-DMJ efficiency, because it makes both the information matrix exactly sparse and the size of the state vector bounded by the number of features.

The paper proposes metrics for quantifying the performance of different mapping algorithms focusing on evaluating their consistency, accuracy and efficiency. The I-DMJ algorithm and a number of existing SLAM algorithms are evaluated using the proposed metrics. The simulation data sets and a preprocessed Victoria Park data set used in this paper are made available to enable interested researchers to compare their mapping algorithms with I-DMJ.

S. Huang, Z. Wang, G. Dissanayake
Faculty of Engineering and Information Technology, ARC Centre of Excellence for Autonomous Systems, The University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia
Tel.: +61-2-95142964
Fax: +61-2-95142655
E-mail: {sdhuang,zhwang,gdissa}@eng.uts.edu.au

U. Frese
University of Bremen, Germany
E-mail: ufrese@informatik.uni-bremen.de

Keywords SLAM · Consistency · Sparse matrix

1 Introduction

Recently, many SLAM algorithms that can efficiently solve large-scale SLAM problems have become available (e.g. see [2] and the references therein). Especially, different techniques exploiting the sparseness of the information matrix of the SLAM problem have been shown to significantly reduce the computational cost. One way to achieve sparseness is to keep all or part of the robot poses in the state vector. This, however, results in an increase in the size of the state vector even when the robot is revisiting the previously observed places, which seems to be unnecessary in an ideal SLAM algorithm [3]. Apart from the computational issue, another even more important issue that is receiving attention in the SLAM community is consistency [4][5]. Recently it has become clear that the major cause of SLAM inconsistency is due to the fact that the Jacobian of observation functions with respect to a feature gets evaluated at different feature location estimates, resulting in the flow of incorrect information to the estimation process [6][7].

This paper makes two main contributions. The first is a new SLAM algorithm for the mapping of large-scale environments by combining local maps – Iterated D-SLAM Map Joining (I-DMJ). The local maps can be generated by any reliable SLAM algorithm (such as Maximum Likelihood (ML)[1] approach). Relationships between the locations of the features in the local map are then extracted and used to build a global map. There are three important features of this algorithm: (1) The global map state vector only contains the feature positions but no robot pose, thus the size of the

[1] also called Smoothing and Mapping (SAM)

state vector does not increase when the robot revisits previously observed places. (2) When joining maps I-DMJ ignores the odometry information connecting successive maps. This makes the information matrix for the global map exactly sparse, leading to significant computational advantages. (3) In the global map building process, the Jacobians with respect to the same feature are evaluated at the same feature location estimate, which improves the consistency of the global map estimate.

The second contribution is on quantitatively evaluating the consistency, accuracy and efficiency of mapping algorithms. Although many SLAM algorithms have been published in the literature, currently there is no systematic way to evaluate their performance. In this paper a number of metrics are proposed and the performance of some existing SLAM algorithms and I-DMJ are evaluated using these metrics.

The paper is organized as follows. Section 2 describes the key features of the I-DMJ algorithm. The details of the I-DMJ algorithm is given in Section 3. In Section 4, some metrics used to compare different mapping algorithms are proposed and discussed. Section 5 uses a number of simulation and experimental examples to compare I-DMJ with some other mapping algorithms. Section 6 concludes the paper.

2 Some key features of I-DMJ algorithm

The I-DMJ algorithm combines the ideas of D-SLAM [1] and Iterated Sparse Local Submap Joining Filter (I-SLSJF) [29] for building feature-only global map. This section lists the key features of the I-DMJ algorithm.

2.1 Obtaining relative information from the local map

The local map provides the position of the local features and the final robot pose in a coordinate frame located at the robot start pose when building the local map. This information is first converted into the relative information among the features, such as the distances and angles among the features. Fig. 1 illustrates the physical meaning of the relative information.

Use of relative information among features makes it possible to formulate the map joining problem where the only variables are the feature positions. Thus robot poses are no longer needed in the global state vector.

2.2 Use of two common features to link the local maps together

As shown in Fig. 2, with at least two common features, the relative information gathered from the local maps

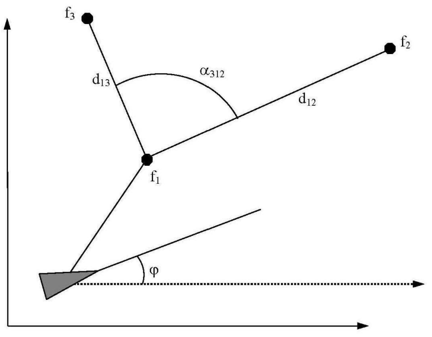


Fig. 1 Relative location information among features f_1, f_2, f_3 – distances d_{12}, d_{13} and angle α_{312}

can be fused together to build a single global map in a common coordinate frame.

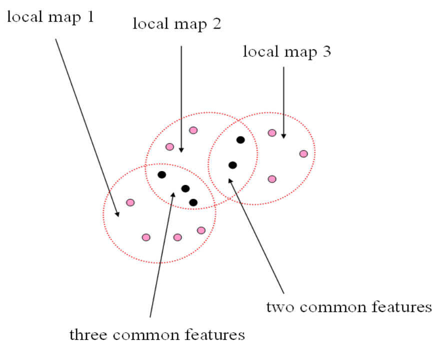


Fig. 2 Use common features to link the local maps together

When two consecutive local maps do not contain at least two common features, "admissible local maps" can be constructed from a set of local maps to satisfy this condition. The details of the process required are given in Section 3.3.

2.3 Formulating map joining as an optimization problem

Since the local map information has been transferred into relative information among features, the map joining problem can be formulated as a least squares problem. The variables to be estimated are the global position of all the features.

An accurate solution to the optimization problem is obtained by combining Extended Information Filter (EIF) with smoothing. Since the Jacobians with respect to the same feature are evaluated at the same feature location estimate, the consistency of the global map estimate is improved.

2.4 Sparse information matrix

Since the map joining problem is a large-scale least squares optimization problem where the information is the relationships among nearby features within a small local map, the information matrix is naturally sparse. Fig. 3 illustrates this.

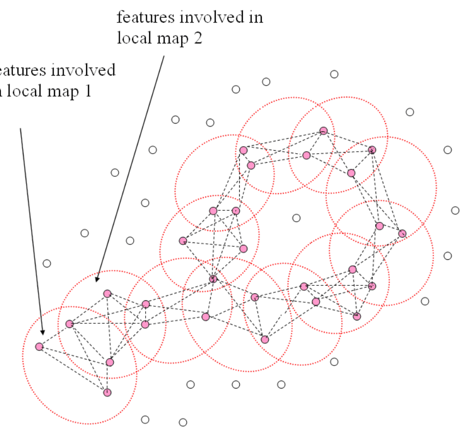

features involved in local map 2
features involved in local map 1

Fig. 3 Each feature is only linked to the features that share the same local map with it.

3 The I-DMJ algorithm

This section provides the details of the I-DMJ algorithm.

3.1 The input and output

The input to the I-DMJ is a sequence of local maps constructed by some SLAM algorithm, which is the same as that of Sparse Local Submap Joining Filter (SLSJF) [27] and I-SLSJF [29]. A local map is 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 of local map k is the same as the robot start pose of local map $k+1$.

The output of I-DMJ is a global map. The global map state vector contains all the feature positions (all the shaded features in Fig. 3). The global map result is given in the form of a global state estimate, an information vector and an information matrix.

3.2 The overall structure of I-DMJ

The overall structure of I-DMJ is presented in Algorithm 1.

Algorithm 1 Overall structure of I-DMJ

```
1: Combine local maps into "admissible" local maps
2: Set local map 1 as the *global map*
3: For $k = 1 : s - 1$ ($s$ is the total number of admissible local
   maps),
   fuse *local map* $k + 1$ into the *global map*
4: End
```

3.3 Combining local maps into "admissible" local maps

The objective of building "admissible" local maps is to make sure there are at least two common features between two consecutive local maps. We therefore check, whether this is already the case. If not, some local maps are combined together (e.g. using I-SLSJF [29]) to construct one new "admissible" local map.

Fig. 4 shows an example of this process. There are less than two common features between local map 1 and local map 2 as well as between local map 2 and local map 3, but there are two common features between local map 3 and local map 4, so local maps 1-3 are combined as a new "admissible" local map 1. The new local map contains all the features involved in local maps 1-3 as well as the final robot pose in local map 3. The coordinate frame of the new local map 1 is the same as that of the old local map 1.

3.4 The steps required for fusing local map into global map

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

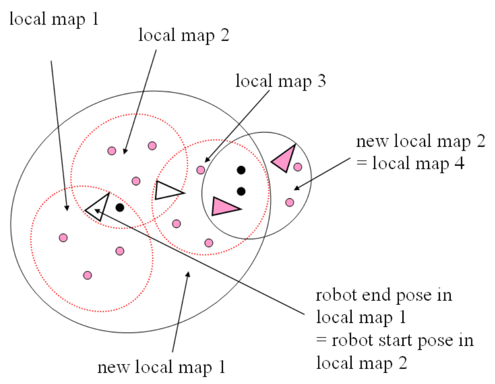


Fig. 4 Local maps 1-3 are combined to form a new "admissible" local map 1.

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

```
1: Convert the local map into relative position information among features
2: Data association
3: Initialization new features using EIF
4: Update using EIF
5: Use least squares to do smoothing when necessary
```

3.5 Obtaining relative information from the local map

Suppose features 1 and 2 in the local map are present in the existing global map. The relative distances and angles from other features with respect to features 1 and 2 can be computed by:

$$z_{rel}^L = \begin{bmatrix} d_{12} \\ \alpha_{312} \\ d_{13} \\ \vdots \\ \alpha_{n12} \\ d_{1n} \end{bmatrix} = \begin{bmatrix} \sqrt{(\hat{x}_2^L - \hat{x}_1^L)^2 + (\hat{y}_2^L - \hat{y}_1^L)^2} \\ \text{atan2}\left(\frac{\hat{y}_3^L - \hat{y}_1^L}{\hat{x}_3^L - \hat{x}_1^L}\right) - \text{atan2}\left(\frac{\hat{y}_2^L - \hat{y}_1^L}{\hat{x}_2^L - \hat{x}_1^L}\right) \\ \sqrt{(\hat{x}_3^L - \hat{x}_1^L)^2 + (\hat{y}_3^L - \hat{y}_1^L)^2} \\ \vdots \\ \text{atan2}\left(\frac{\hat{y}_n^L - \hat{y}_1^L}{\hat{x}_n^L - \hat{x}_1^L}\right) - \text{atan2}\left(\frac{\hat{y}_2^L - \hat{y}_1^L}{\hat{x}_2^L - \hat{x}_1^L}\right) \\ \sqrt{(\hat{x}_n^L - \hat{x}_1^L)^2 + (\hat{y}_n^L - \hat{y}_1^L)^2} \end{bmatrix} \tag{3}$$

The physical meaning of the distances and angles is shown in Fig. 1.

The corresponding covariance matrix of the noise on these relative information, P_{rel}^L, can be computed by the Jacobian of the relationship between the local map and the relative information (3), together with the covariance matrix of the local map (P^L in (1)). The process is the same as that of D-SLAM Map Joining (DMJ) [18].

3.6 Data association

Data association here refers to finding those features present in local map $k+1$ that are already included in the global map and their corresponding indices in the global state vector.

Due to the admissibility condition for local maps, there are at least two features in the local map that are already included in the global map. Hence, the local map and the global map can be transferred into the same coordinate system and the data association can be performed in a way similar to the data association in SLSJF [27]. Note that one of the key steps is to recover the covariance sub-matrix corresponding to the possibly matched features.

When the estimation error is too large (e.g. just before closing a large loop), some global localization technique (such as [28]) is necessary for finding the match between the local map and the global map.

3.7 Local map fusion as a least squares problem

The relative information (z_{rel}^L, P_{rel}^L) can be treated as a measurement of the true relative positions among the features with a zero-mean Gaussian noise whose covariance matrix is P_{rel}^L. Computing a global map from this information is just a standard non-linear least square problem. It is linearized and converted into a (sparse) linear equation system and then solved with (sparse) Cholesky decomposition [26, §2.7].

To state it clearly, suppose the features involved in the local map are $X_1^G = (x_1^G, y_1^G), \cdots, X_n^G = (x_n^G, y_n^G)$, then the relative information of the local map z_{rel}^L can be regarded as a measurement of the true relative positions among the features $X_1^G, \cdots, X_n^G$. That is,

$$z_{rel}^L = H(X^G(k)) + w_{map} \tag{4}$$

where $X^G(k)$ is the global state vector after fusing local map k, and $H(X^G(k))$ is the vector of relative positions given by

$$H(X^G) = \begin{pmatrix} \sqrt{(x_2^G - x_1^G)^2 + (y_2^G - y_1^G)^2} \\ \text{atan2}\left(\frac{y_3^G - y_1^G}{x_3^G - x_1^G}\right) - \text{atan2}\left(\frac{y_2^G - y_1^G}{x_2^G - x_1^G}\right) \\ \sqrt{(x_3^G - x_1^G)^2 + (y_3^G - y_1^G)^2} \\ \vdots \\ \text{atan2}\left(\frac{y_n^G - y_1^G}{x_n^G - x_1^G}\right) - \text{atan2}\left(\frac{y_2^G - y_1^G}{x_2^G - x_1^G}\right) \\ \sqrt{(x_n^G - x_1^G)^2 + (y_n^G - y_1^G)^2} \end{pmatrix} \tag{5}$$

and w_{map} is the zero-mean measurement noise whose covariance matrix is P_{rel}^L.

So the problem of fusing the admissible local maps 1 to k is to estimate the global state $X^G(k)$ using all the local map information (4) for $j = 1, \cdots, k$. This problem can be formulated as a least squares problem:

$$\min_{X^G(k)} \sum_{j=1}^{k} (z_{rel}^{Lj} - H_j(X^G(k)))^T (P_{rel}^{Lj})^{-1} \\ (z_{rel}^{Lj} - H_j(X^G(k))), \tag{6}$$

where $(z_{rel}^{Lj}, P_{rel}^{Lj})$ is the relative information computed from local map j, and H_j is the function in (5) applied to local map j.

The least squares problem can be solved iteratively. In fact, the following linear equation can be used to compute the new estimate $\hat{X}_{new}^G(k)$ when the previous estimate $\hat{X}_{old}^G(k)$ is available:

$$\left[\sum_{j=1}^{k} \nabla H_j^T (P_{rel}^{Lj})^{-1} \nabla H_j\right] \hat{X}_{new}^G(k) \\ = \sum_{j=1}^{k} \nabla H_j^T (P_{rel}^{Lj})^{-1} [z_{rel}^{Lj} - H_j(\hat{X}_{old}^G(k)) + \nabla H_j \hat{X}_{old}^G(k)] \tag{7}$$

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

In (7), the matrix

$$I(k) = \sum_{j=1}^{k} \nabla H_j^T (P_{rel}^{Lj})^{-1} \nabla H_j \tag{8}$$

is called information matrix. The vector

$$i(k) = \sum_{j=1}^{k} \nabla H_j^T (P_{rel}^{Lj})^{-1} [z_{rel}^{Lj} - H_j(\hat{X}_{old}^G(k)) \\ + \nabla H_j \hat{X}_{old}^G(k)] \tag{9}$$

is called information vector. Note that the information matrix is an exactly sparse matrix, so the state estimate $\hat{X}_{new}^G(k)$ can be obtained by solving a sparse linear equation, which can be done efficiently by sparse Cholesky decomposition.

3.8 Initialize the unmatched features

The initial values of the global positions of all unmatched features in local map $k+1$ are computed (using z_{rel}^L and the estimate of the global position of the two common features) and inserted to $\hat{X}^G(k)$. The dimensions of $i(k)$, $I(k)$ are increased by adding zeros corresponding to the new added features.

3.9 Update the global map using EIF

The process of updating the global map using EIF is stated in Algorithm 3. The details is similar to those of SLSJF [27] and I-SLSJF [29] and is omitted here.

Algorithm 3 Update using EIF

```
1: Compute the information matrix and information vector using EIF
2: Reorder the global map state vector when necessary
3: Compute the Cholesky Factorization of $I(k+1)$
4: Recover the global map state estimate $\hat{X}^G(k+1)$
```

3.10 Smoothing by iteration using least squares

Whenever the change of state estimate computed by the EIF is larger than a predefined threshold, a smoothing step is performed. The steps for smoothing using the least squares method are listed in Algorithm 4.

Algorithm 4 Smoothing using least squares

```
1: Recompute the information matrix $I(k+1)$ and the information vector $i(k+1)$
2: Compute the Cholesky Factorization of $I(k+1)$
3: Recover the global map state estimate $\hat{X}^G(k+1)$
4: Repeat the above process until $\hat{X}^G(k+1)$ converges.
```

When recomputing the information matrix $I(k+1)$ and the information vector $i(k+1)$, the most recent global state estimate is used.

3.11 Pros and cons of I-DMJ

As compared with ML, the dimension of the state vector in I-DMJ only depends on the number of features and thus is significantly lower. Moreover, the number of global map update operations in I-DMJ is equal to the number of local maps which is only a small fraction of the total number of time steps. In addition, the exact sparseness of the information matrix is still maintained, so the I-DMJ algorithm is computationally very efficient. An obvious question to pose is “what is sacrificed in the proposed I-DMJ algorithm”?

In I-DMJ, the local maps are first transferred into the relative positions information among the local map features, then the relative information is treated as an integrated measurement in the global map building. By doing so, all the features within the same local map have links with each other in the information matrix, where as these feature-to-feature links do not exist in the ML information matrix. That is, while still sparse, the information matrix in I-DMJ is "denser" than that in ML.

More importantly, I-DMJ is actually solving a slightly different optimization problem as compared with the optimization problem solved by ML. When the Gaussian relative information (z_{rel}^L, P_{rel}^L) consistently represent the local map feature relationships, it can be argued that the new optimization problem formulated in I-DMJ is a simplified version of the original ML optimization problem. However, both the local map building process and coordinate transformation process involve linearization. Because of the linearization, it may be inaccurate to use Gaussian distribution (z_{rel}^L, P_{rel}^L) to represent the relative information of the local map.

Furthermore, in the I-DMJ algorithm, even though the Jacobians are re-evaluated in computing the global map, the Jacobians used to compute P_{rel}^L are never reevaluated. Although it can be argued that the effect of this source of linearization error should be minor because the local maps are all with small sizes, the potentially inconsistency due to this still exists.

Since no robot pose is present in the state vector of I-DMJ, the odometry information corresponding to the robot's pose change between consecutive local maps is not used in the map joining. This information loss may result in less accurate estimate as compared with ML. Because most of the odometry information has already been exploited in the local map building process, this information loss is relatively small compared to all the information available, especially when the observation sensor is of good quality and with a high sensing rate.

Overall, there will be some difference between the I-DMJ solution and the ML map solution. How large the difference is mainly depends on the quality of the local maps. To improve the quality of the local maps, we recommend to use ML to build them (and then marginalize out the previous robot poses).

In I-DMJ, the robot poses in the global coordinate frame are not estimated. If it is necessary to estimate the global robot trajectory, the localization process in D-SLAM [1] can be used to obtain an estimate of the coordinate frames of the local maps. Combining the coordinate frames with the local robot trajectory (e.g. from the ML local map), an estimate of the global robot trajectory can be obtained. Alternatively, an estimate of the robot trajectory can be computed by solving a least squares problem with all the feature positions fixed at the I-DMJ estimate [2].

As compared to DMJ [18], I-DMJ performs some extra smoothing steps aiming to produce more consistent estimates. So the computational cost of I-DMJ is higher than that of DMJ.

In summary, like many other SLAM algorithms, there are some tradeoffs among consistency, accuracy and efficiency in the proposed I-DMJ algorithm. In the next section, we will propose some metrics for quantifying the consistency, accuracy and efficiency of SLAM algorithms such that the tradeoffs of different SLAM algorithms can be compared.

4 Metrics for quantifying the performance of mapping algorithms

Quantitatively comparing the performance of different mapping algorithms is extremely important for future deployment of mapping in real applications. The performance of an algorithm covers a broad range of aspects. In this paper, we only focus on the performance in terms of consistency, accuracy and efficiency for point feature based SLAM algorithms.

4.1 Consistency

For SLAM algorithms, estimate consistency is arguably more important than the computational efficiency of the algorithm. A good SLAM algorithm should always provide some sort of confidence level (such as the covariance) on the estimate computed so that the outcomes can be used effectively in making decisions while giving due consideration to the errors these contain. An inconsistent result can be either overconfident or conservative in terms of the confidence level. It could be argued that overconfident estimates are more harmful than conservative estimates as the resulting decisions based on an overconfident result may lead to mission failure.

4.1.1 A fundamental cause of SLAM inconsistency

Recently, the SLAM consistency issue has been revisited by a number of researchers. It has become clear that the most significant cause of overconfident estimate for a SLAM algorithm is due to the fact that the Jacobian of observation/odometry functions with respect to the same feature/pose gets evaluated at different feature/pose location estimates, resulting in the flow of incorrect information to the estimation process [6][7]. Some examples are given in the following to explain this.

[2] This problem is solved in Section 4.2.2 to compare the accuracy of the experimental results of different mapping algorithms.

Sequential update vs batch update of the EKF SLAM. Assume that the robot can observe more than one features at some observation points and the sequential update uses the observations one by one to update the state estimate. Instead, the batch update strategy uses all the observations made at the same observation point as an integrated observation and updates the state only once. During sequential updates, the estimate of the current robot pose (the observation point) is changed slightly after each update step, so the Jacobian with respect to the current robot pose is evaluated at slightly different values at each update step. This makes the EKF with sequential update overconfident. Fig. 6(a) and Fig. 6(b) show that the behaviour of the two algorithms is significantly different.

This is remarkable, since with sequential EKF Jacobians are evaluated with a later, i.e. more precise estimate as linearization point. The implication is that evaluating the Jacobian at the same linearization point is more important than evaluating it at a good linearization point [7].

EKF SLAM vs ML. In the EKF algorithm (using batch update), when the observations made from different poses to the same feature are used to update the estimate, the Jacobians with respect to the same feature are evaluated at different estimate values. Since the previous poses are not included in the traditional SLAM state vector, this issue cannot be avoided and thus EKF performs worse than ML, in which all the robot poses are included in the state vector and the Jacobians can be re-evaluated at any time step. Fig. 7(a) and Fig. 7(b) show the large difference between the performance of the two algorithms.

SLSJF vs I-SLSJF. In SLSJF [27], a single EIF is used for the global map estimation and the Jacobians are never recomputed, so the same issue arises. I-SLSJF [29] overcomes this issue by recomputing the Jacobians and information matrices whenever it is found necessary. The examples in [29] show the improvement of I-SLSJF over SLSJF.

DMJ[18] vs I-DMJ. Similar to the above, I-DMJ does smoothing and iterations by recomputing the Jacobians whenever necessary and thus performs better than the DMJ algorithm. Fig. 8(b) and Fig. 8(d) show that the behaviour of the two algorithms is different.

4.1.2 Quantification of estimation consistency in simulation

The consistency of an algorithm can be examined in simulation by performing Monte Carlo runs. For example, it is possible to perform a large number of simulations with independent sensor noises and use the resulting state estimates as samples to generate the error covariance. Comparing this covariance with the one obtained from the mapping algorithm will give us a quantitative measure of the consistency of the mapping algorithm. The comparison between the two covariances can be performed by calculating the generalized eigenvalue spectrum of one covariance matrix relative to the other [3]. Another way to quantify the consistency is to compute the average normalised estimation error squared (NEES) of different runs [4], and then perform a χ^2 (chi-square) test. More discussion on these two approaches are given in the following sections. In Section 5, we use a small simulation data set to demonstrate the two approaches.

For large-scale problems where performing Monte Carlo simulations is very time consuming, a simpler (but inadequate because the error sequence is correlated [4]) consistency check can be performed by examining the sequence of NEES over a single simulation run. Although this is not adequate for consistency check of a particular algorithm, it may still be used to compare the consistency of different algorithms. In Section 5, the consistency of different algorithms are compared using this strategy with some larger simulation data sets.

4.1.3 Comparison of covariances using generalized eigenvalues

The NEES is a single number which should have a $\chi^2(2N)$ distribution (where N is the number of features in the map) and allows us to assess the consistency of a $2N$-dimensional map estimate. However, it combines the error in all $2N$ dimensions into one number. Thus NEES is not able to adequately deal with the scenario where some aspect of the estimate was overconfident and some aspect was conservative.

Is there a way to assess the consistency in *all* aspects of the map? There is [3], but more data is needed to do so. The algorithm must be run many times with independent measurement noise, so the error covariance of the algorithm's estimate can be estimated by Monte Carlo simulation as

$$P_{MC} = \frac{1}{r}\sum_{l=1}^{r}(\hat{x}_l - x)(\hat{x}_l - x)^T \tag{10}$$

where r is the number of Monte Carlo runs, $\hat{x}_l$ is the estimate from the l-th run and x is the ground truth. Ideally, $\hat{x}_l$ and x should be the estimate and the ground truth of the whole map X^G but then too many Monte Carlo runs are necessary to approximate the error covariance of the estimate of the $2N$-dimensional map.

So in the simulation example in Section 5.1.1, x only represents the positions of 3 selected features.

Let P be the average of the algorithm's own covariance given by

$$\bar{P} = \frac{1}{r}\sum_{l=1}^{r} P_l \tag{11}$$

where P_l is the covariance of the l-th run.

Now for every direction g in state space, i.e. every aspect of the map, we are interested in the actual error $\sqrt{g^T P_{MC} g}$ of the algorithm as compared to the estimated error $\sqrt{g^T \bar{P} g}$. If the ratio

$$\mathrm{ruc}(g) = \sqrt{\frac{g^T P_{MC} g}{g^T \bar{P} g}} \tag{12}$$

is > 1, then the algorithm is overconfident in direction g; if the ratio is < 1, then the algorithm is conservative in direction g. Now, the key point is, that the value ruc(g) for different g can be characterized by the so-called generalized eigenvalues λ_i and eigenvectors v_i $(1 \le i \le 2N)$, defined by

$$P_{MC} v_i = \lambda_i \bar{P} v_i. \tag{13}$$

These have the following properties

$$v_i^T P_{MC} v_i = \lambda_i, \quad v_i^T \bar{P} v_i = 1, \quad \forall i, \tag{14}$$

$$v_i^T P_{MC} v_j = 0, \; v_i^T \bar{P} v_j = 0, \quad \forall i \ne j. \tag{15}$$

This means, the v_i are uncorrelated directions both in P_{MC} and $\bar{P}$ and have $\mathrm{ruc}(v_i) = \sqrt{\lambda_i}$ as error ratio. Hence the spectrum of generalized eigenvalues λ_i characterizes the consistency in *every* aspect of the map in a very concise way.

With the same method one can also compare the error of one algorithm relative to another algorithm in all aspects of the map, simply by considering the covariances obtained by (10) for both algorithms.

4.1.4 Relation between NEES and generalized eigenvalues

The question arises, whether there is a relation between NEES as a one-number indicator of consistency and the λ_i. Indeed, roughly the sum $\sum_{i=1}^{2N} \lambda_i$ equals the average NEES. The term "roughly" corresponds to the following: Normally, if one conducts r Monte-Carlo runs of an algorithm one would compute the average NEES as

$$\overline{\mathrm{NEES}} = \frac{1}{r}\sum_{l=1}^{r} (\hat{x}_l - x)^T P_l^{-1} (\hat{x}_l - x), \tag{16}$$

i.e. would use the l-th estimate $(\hat{x}_l)$ and l-th covariance (P_l) to compute the l-th NEES. However, while the estimation errors are completely different each run, P_l should be roughly the same, since it would be exactly the same in the linear case. So (16) is approximately

$$\overline{\mathrm{NEES}} \approx \frac{1}{r}\sum_{l=1}^{r} (\hat{x}_l - x)^T \bar{P}^{-1} (\hat{x}_l - x), \tag{17}$$

with $\bar{P}$ given by (11).

Now we prove that this approximate of $\overline{\mathrm{NEES}}$ is equal to the sum of the generalized eigenvalues, that is

$$\frac{1}{r}\sum_{l=1}^{r} (\hat{x}_l - x)^T \bar{P}^{-1} (\hat{x}_l - x) = \sum_{i=1}^{2N} \lambda_i. \tag{18}$$

Let $LL^T = \bar{P}$ be the Cholesky-decomposition of $\bar{P}$. Then $L^T v_i$ are eigenvectors of $L^{-1} P_{MC} L^{-T}$, since

$$\begin{aligned} (L^{-1} P_{MC} L^{-T})(L^T v_i) &= L^{-1} P_{MC} v_i \\ &= \lambda_i L^{-1} \bar{P} v_i \\ &= \lambda_i L^{-1} L L^T v_i \\ &= \lambda_i (L^T v_i). \end{aligned} \tag{19}$$

So, $\sum_{i=1}^{2N} \lambda_i = \mathrm{tr}(L^{-1} P_{MC} L^{-T})$. With this we can conclude, that

$$\begin{aligned} &\tfrac{1}{r}\textstyle\sum_{l=1}^{r} (\hat{x}_l - x)^T \bar{P}^{-1} (\hat{x}_l - x) \\ &= \tfrac{1}{r}\textstyle\sum_{l=1}^{r} (\hat{x}_l - x)^T L^{-T} L^{-1} (\hat{x}_l - x) \\ &= \tfrac{1}{r}\textstyle\sum_{l=1}^{r} \left(L^{-1}(\hat{x}_l - x)\right)^T \left(L^{-1}(\hat{x}_l - x)\right) \\ &= \tfrac{1}{r}\,\mathrm{tr}\left(\textstyle\sum_{l=1}^{r} \left(L^{-1}(\hat{x}_l - x)\right)\left(L^{-1}(\hat{x}_l - x)\right)^T\right) \\ &= \tfrac{1}{r}\,\mathrm{tr}\left(L^{-1}\left(\textstyle\sum_{l=1}^{r} (\hat{x}_l - x)(\hat{x}_l - x)^T\right) L^{-T}\right) \\ &= \mathrm{tr}\left(L^{-1} P_{MC} L^{-T}\right) \\ &= \textstyle\sum_{i=1}^{2N} \lambda_i. \end{aligned} \tag{20}$$

In Tables 2, both the average NEES and the sum of the eigenvalues are presented to show that they are "roughly" equal.

4.1.5 Quantification of estimation consistency in experimental results

Evaluating the consistency of an estimator on an experimental data set is almost impossible. One reason is that ground truth is normally not available. Another reason is that it is almost impossible to repeat the same experiment (with the same trajectory but different independent noise) many times.

Although a proper quantification of algorithm consistency using a single experimental data is unlikely to be available, it is possible to perform some checks on the estimation results to provide some indication of whether the estimate might be consistent or not. These are discussed in the following.

If the ground truth of the feature positions is available, then the sequence of NEES over the different steps can be used to perform a consistency check. When the ground truth of the feature positions is not available but some other ground truth information related to the true feature positions is available, some consistency checks are possible. Two examples are given below.

Case 1: the relative information among a few features (together with the associated uncertainties) is available. This information can be obtained by, for example, measuring the distances between these features by hand.

The relative information Z is a function of the state vector X (the state vector X of a mapping algorithm may also contain some robot poses but it should contain all the feature positions). We use $F(X)$ to express this function. Denote the relative information available as $\hat{Z}$ with a covariance matrix P_Z representing its uncertainty. Then the information available can be expressed as

$$F(X) \sim N(\hat{Z}, P_Z). \tag{21}$$

From the estimation result of the algorithm, the state estimate $\hat{X}$ and the corresponding covariance matrix P_X, we can get an estimate of $F(X)$, $F(\hat{X})$, together with a covariance matrix P_{ZX}. One simple way to compute P_{ZX} is using $P_{ZX} = F_X P_X F_X^T$ where F_X is the Jacobian of the function F with respect to X evaluated at $\hat{X}$.

Suppose the mapping algorithm is consistent, then

$$X \sim N(\hat{X}, P_X) \tag{22}$$

and

$$F(X) \sim N(F(\hat{X}), P_{ZX}) \tag{23}$$

provided that $\hat{X}$ is close enough to X and the linearization is applicable.

From (23) and (21), we can perform a χ^2 (chi-square) test by computing

$$(Z - F(\hat{X}))^T (P_Z + P_{ZX})^{-1} (Z - F(\hat{X})) \tag{24}$$

and comparing it with the 99% (or 95%) probability concentration region of a χ^2 distribution with $\dim(F(X))$ degrees of freedom.

Case 2: true data associations between some measurements are available. In this case, the correct data association information can be used for a consistency check.

Suppose the correct data association tells that f_1 and f_2 are the same feature. We can first implement a mapping algorithm assuming f_1 and f_2 are two different features and obtain the state estimate $\hat{X}$ and the corresponding covariance matrix P_X. The difference between the two feature positions can be expressed as a function of the state vector,

$$d = G(X) = X_{f_1} - X_{f_2},$$

where X_{f_1} and X_{f_1} denote the positions of features f_1 and f_2, respectively. So an estimate of d and the associated covariance matrix can be obtained by

$$\hat{d} = \hat{X}_{f_1} - \hat{X}_{f_2}, \quad P_d = G_X P_X G_X^T, \tag{25}$$

where G_X is the Jacobian of G evaluated at $\hat{X}$.

Suppose the mapping algorithm is consistent, then because the true value of d is 0, then

$$\hat{d} P_d^{-1} \hat{d}^T = (\hat{X}_{f_1} - \hat{X}_{f_2})^T P_d^{-1} (\hat{X}_{f_1} - \hat{X}_{f_2}) \tag{26}$$

should pass the χ^2 test with 2 degrees of freedom (for 2D feature mapping).

This approach can be generalized to using the true data association of a few pairs of features for a consistency check.

In summary, any ground truth information related to the state vector can be used to perform a check on the algorithm consistency. Obviously, passing the check does not mean the algorithm is absolutely consistent.

Please note that for both the above two cases, the whole covariance P_X is not necessary for the consistency check. It is only required that covariance submatrix corresponding to the related features is available. Thus the approaches can be applied to information filter based algorithms where the whole covariance matrix is not maintained (efficient covariance submatrix recovery techniques are available in [16] [27], for example).

4.2 Accuracy

4.2.1 Quantification of estimation accuracy in simulation

If both algorithms are consistent, then the accuracy comparison can be done by comparing the covariance matrices produced by the two algorithms. In simulation, the best way is to perform Monte Carlo runs and compare the P_{MC} in (10) from the two algorithms. The two covariance matrices can be properly compared by computing the generalized eigenvalues. As “the minimal uncertainty that could be theoretically derived from the information available” is from the optimal ML estimate, it is reasonable to treat the ML covariance matrix as a benchmark [3].

4.2.2 Quantification of estimation accuracy in experiments

The accuracy comparison of different algorithms using only one experimental data is extremely difficult (and even impossible) when the ground truth of the feature positions is not available, mainly because that the consistency can not be checked properly.

When the ground truth of data association and the statistical properties of the sensor noises on odometry and observations are available, it can be argued that the best solution one can achieve is the ML solution that minimizes the χ^2 error defined by

$$\begin{aligned}\chi^2(X) &= \chi^2\begin{pmatrix}X_f\\X_r\end{pmatrix}\\ &= \sum_{i=1}^{m}(Z_i - H_{Zi}(X))^T P_{Zi}^{-1}(Z_i - H_{Zi}(X))\\ &\quad+\sum_{j=1}^{p}(O_j - H_{Oj}(X))^T P_{Oj}^{-1}(O_j - H_{Oj}(X))\end{aligned} \tag{27}$$

where X_f denotes all the feature positions and X_r denotes all the robot poses, Z_i $(1 \le i \le m)$ are observations and O_j $(1 \le j \le p)$ are odometries. P_{Zi} and P_{Oj} are the corresponding covariance matrices; H_{Zi} and H_{Oj} are the corresponding functions relating them to the state X.

Since on the one hand this minimum, i.e. the ML solution is on average the best estimate and on the other hand, any other estimate $\hat{X}$ has by definition a higher χ^2 error, one could view the difference

$$\Delta\chi^2(\hat{X}) = \chi^2(\hat{X}) - \min_X \chi^2(X) \tag{28}$$

as an indicator of how much worse $\hat{X}$ is compared to the ML solution.

According to [26, §15.6, Theorem C], $\Delta\chi^2(\hat{X})$ is distributed as a chi-square distribution with $\dim(X)$ degree of freedom. So the ratio

$$ratio = \frac{\Delta\chi^2(\hat{X})}{\dim(X)} \tag{29}$$

is a rough indicator of the error of $\hat{X}$ relative to ML, given that the assumptions underlying ML hold.

The discussion so far covers estimates such as ML, that include all features and robot poses. Often we have an estimate $\hat{X}_f$ without robot poses. Then the robot poses X_r have to be estimated by minimizing the χ^2 error and $\dim(X)$ is replaced by $\dim(X_f)$ [26, §15.6, Theorem D]. The relevant ratio to be used in that case is

$$error\ \ ratio = \frac{\min_{X_r}\chi^2\begin{pmatrix}\hat{X}_f\\X_r\end{pmatrix} - \min_X \chi^2(X)}{\dim(X_f)}. \tag{30}$$

4.3 Efficiency

Currently many efficient SLAM algorithms are based on exploiting the sparseness of the information matrix. Some common techniques proposed include factorization (either QR or Cholesky) [8], incrementally construction of factorization [16][27], reordering of state vector [8][27], divided and conquer [15] or tree representation [21][30], etc.

Although some algorithm claims $O(n)$ or $O(\log n)$ computational cost, the meaning of n varies from algorithm to algorithm and the constant involved might be extremely large for some algorithms. Moreover, it can be argued that it is impossible to achieve $O(n)$ computational cost in some worse case scenarios (e.g. “mowing the lawn” case) due to the general $O(n^{1.5})$ cost of planar grids [20].

Since the actual computational time also depends on how the algorithm is implemented and optimized, sometimes it does not make too much sense to simply compare the execution time of different algorithms.

For SLAM algorithms that exploit sparse information matrices, both the computational cost and storage requirement highly depends on the number of non-zero elements in the information matrix and the times these non-zero elements are accessed in the algorithm. So in this paper, it is proposed that (i) the number of non-zero elements in the final information matrix, and (ii) the total times of access to the non-zero elements of the information matrices, be used as two indicators of the efficiency of an algorithm.

Table 1 summarizes some key factors that affect the number of non-zero elements in the information matrix and the number of times the non-zero elements are accessed by various SLAM algorithms. It should be noted that by incrementally constructing the information matrix (or its factorization), the total number of times the non-zero elements are accessed is significantly reduced. However, this is also one of the major causes of the inconsistency as pointed out in Section 4.1.1.

5 Simulation and experiment results

In this section, simulation and experiment results are provided to compare a number of SLAM algorithms

Table 1 Summary of the key factors influencing the number of nonzero elements in the information matrix and the number of times the non-zero elements are accessed for various SLAM algorithms: (N: number of features, p – number of poses, s – number of local maps)

Algorithms	state dimension	key factors influencing the number of non-zeros	key factors influencing the times of access
ML[8]	$2N+3p$	p, number of observations	number of iterations
iSAM [16],	$2N+3p$	p, number of observations	incrementally build up factorization
I-SLSJF [29]	$2N+3s$	s, number of features in the local maps	number of smoothing steps
SLSJF [27]	$2N+3s$	s, number of features in the local maps	incrementally build up information matrix
I-DMJ	$2N$	number of features in the local maps	number of smoothing steps
DMJ [18]	$2N$	number of features in the local maps	incrementally build up information matrix
D-SLAM [1]	$2N$	number of features observed at the same pose	incrementally build up information matrix
ESEIF [9]	$2N+3$	number of features observed at the same pose, limit on the number of "active features" ([9])	incrementally build up information matrix

including I-DMJ in terms of consistency, accuracy and efficiency using the metrics proposed in the previous section.

5.1 Simulation results

Although experimental implementation and evaluation is an essential element of any practical robotic algorithm, simulation is arguably more suitable for evaluating the consistency of different SLAM algorithms because of the availability of the ground truth and the feasibility of repeating the simulation as many times as required. [3]

5.1.1 Simulation with 535 *time steps*

This small data set is used to perform a comparison of the consistency and accuracy of different algorithms.

The small simulation environment contains 196 nearly uniformly distributed features. The robot starts from the bottom-left corner of the square and finishes at the top-right corner (no loop closure) as shown in Fig. 5. The range-bearing sensor is assumed to have a field of view of 180 degrees.

A sensor range of 3 meters was first used. Five thousand simulation data sets were generated (each with the same parameters but different random seeds for the noises). Five different algorithms (EKF with sequential update, EKF with batch update, I-DMJ, I-SLSJF, and ML) were applied to the 5000 data sets and 5000 estimates were generated by each algorithm. For I-DMJ and I-SLSJF, five local maps were first built by EKF and then fused together.

[3] For the simulation data sets mentioned in this paper, the odometry and observation data, the robot motion model, and the parameters used in our algorithms are all available on the website: http://services.eng.uts.edu.au/~sdhuang/research.htm. We hope these data sets can be used by many researchers to evaluate their mapping algorithms before experimental implementations.

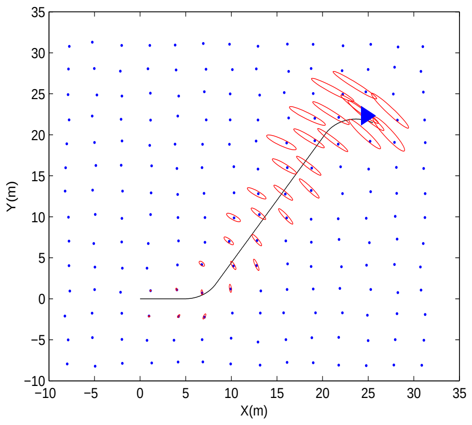


Fig. 5 EKF result of one of the 5000 runs – 535 step simulation data set

For each algorithm, the 5000 estimates were used to compute the P_{MC} in (10). This value of P_{MC} was compared with the average covariance matrix $\bar{P}$ in (11) generated from the same algorithm for evaluating the consistency. The first half of Table 2 shows the results of the comparison using generalized eigenvalues. It can be seen that EKF with sequential update is obviously inconsistent with the maximal eigenvalue $122.9897 \gg 1$. EKF with batch update is also over confident with maximal eigenvalue $3.2304 > 1$. While I-DMJ, I-SLSJF and ML are much better in terms of consistency.

For the accuracy comparison, the P_{MC} from each algorithm is compared to the P_{MC} of ML using the generalized eigenvalue approach. First half of Table 3 shows the results. It is clear that the accuracy of both I-DMJ and I-SLSJF is acceptable because the minimal eigenvalues and the maximal eigenvalues are all close to 1.

Second half of Table 2 and Table 3 show the corresponding results when the sensor range is increased to $3.5m$ with all the other parameters kept the same.

Table 2 The consistency check for different algorithms using the 535 step simulation data. EKF(S) stands for EKF with sequential update; EKF(B) stands for EKF with batch update. λ_i is the generalized eigenvalue of P_{MC} relative to $\bar{P}$ (of the 3 selected features, one near the robot start point, one near the middle, one near the end point). The upper half table is the results when sensor range is 3m while the lower half table is the results when sensor range is 3.5m.

3m	EKF(S)	EKF(B)	I-DMJ	I-SLSJF	ML
$\max\lambda_i$	122.9897	3.2304	1.3301	1.2916	1.0417
$\min\lambda_i$	1.0098	0.9998	0.9640	0.9592	0.8657
$\sum\lambda_i$	150.7355	9.5862	6.4683	6.4195	5.8759
$\overline{\text{NEES}}$	238.2530	12.0948	9.2466	9.1547	8.1803
3.5m					
$\max\lambda_i$	1.5697	1.3891	1.0611	1.0612	1.0471
$\min\lambda_i$	1.0008	0.9883	0.9504	0.9507	0.9410
$\sum\lambda_i$	6.7442	6.5083	6.0558	6.0793	5.9825
$\overline{\text{NEES}}$	7.0599	6.6494	6.2194	6.2426	6.1168

Table 3 The accuracy check for different algorithms using the 535 step simulation data. λ_i is the generalized eigenvalue of P_{MC}^{ML} relative to $P_{MC}^{algorithm}$ (of the 3 selected features). The upper/lower half of the table is the result when sensor range is 3m/3.5m.

3m	EKF(S)	EKF(B)	I-DMJ	I-SLSJF	ML
$\max\lambda_i$	1.0000	1.0028	1.0079	1.0073	1
$\min\lambda_i$	0.0437	0.6142	0.8105	0.8459	1
$\sum\lambda_i$	3.6492	5.1891	5.6283	5.7056	6
3.5m					
$\max\lambda_i$	1.0030	1.0022	1.0019	1.0263	1
$\min\lambda_i$	0.7522	0.8409	0.8933	0.9348	1
$\sum\lambda_i$	5.6431	5.7791	5.7901	5.9384	6

In this case, most of the algorithms provide reasonable results. The consistency of the EKF with sequential update is much better because the maximal λ_i is 1.5697. Comparing the P_{MC} obtained from different algorithms with that from ML, we can see that the accuracy of different algorithms are also acceptable.

The reason why this small increase in sensor range improves the performance significantly is due to the fact that: (1) the turn rate error is relatively large, with variance of 4 degrees per second, (2) the feature density is around $3m$ each feature. For EKF with $3m$ sensor range, the robot can only observe one feature in many of the steps (850 feature observations in 535 steps), thus the accumulated process noise is not able to be reduced by observations. But when the sensor range is $3.5m$, the robot can observe at least two features at each step (1130 feature observations in 535 steps), which is very helpful in locating the robot. For map joining algorithms, the accumulated process noises in the local maps is limited and the smoothing and iteration further reduce the effects of noisy controls.

5.1.2 More simulation results

More simulations were conducted to evaluate the performance of different SLAM algorithms. Table 4 summarizes the different data sets used for the simulations.

Table 5 shows the results of the consistency evaluation for different algorithms using the different data sets by examining the sequence of NEES over the simulation run. Here a pass/fail means the NEES of the final estimate is smaller/larger than the corresponding (one-sided) 99% confidence gate. The total number of passes and the total number of evaluations (one evaluation per step) are shown in the parentheses.

The 3433 time steps data set shows that EKF using batch update performs better than sequential update in terms of consistency. Fig. 6(a) shows the map generated by EKF with sequential update while Fig. 6(b) shows the map generated by EKF with batch update.

The 8240 time steps data set shows that EKF using batch update may fail but map joining algorithms I-DMJ and I-SLSJF perform adequately. Fig. 7(a) and Fig. 7(b) show the maps generated by EKF with batch update and ML, respectively. The results of I-DMJ and I-SLSJF are similar to that of ML and are not shown.

The 35187 time steps data set shows that the I-DMJ and I-SLSJF can efficiently produce consistent and accurate maps for very large-scale SLAM problems. However, without smoothing and iteration, both the DMJ and SLSJF algorithms fail for this data set. Fig. 8(a) shows the map generated by DMJ while Fig. 8(c) shows the map generated by I-DMJ. Close examination of the maps show that DMJ estimates are over confident.

Table 8 compares the non-zero elements in the final information matrix and the total times the non-zero elements have been accessed in different algorithms using the different data sets. This gives an indication of the efficiency of different algorithms.

5.2 Experimental results using DLR-Spatial-Cognition data set

The DLR-Spatial-Cognition data set [4] is collected using a robot equipped with a camera. The robot was moved around in the building with artificial landmarks (white/black circles) placed on the ground. The image data has been preprocessed and the relative position of the observed landmarks with respect to the observation point are provided. The odometry information is also available from the data set in the form of relative poses between every two consecutive poses. The

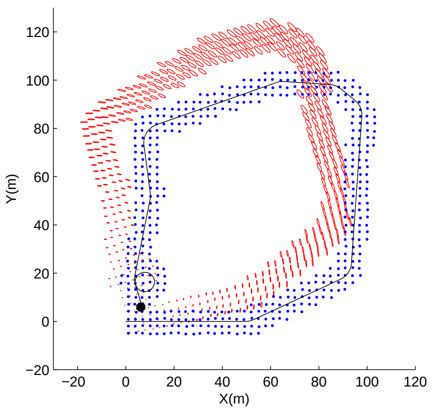
[4] The data set is available at *http://www.sfbtr8.spatial-cognition.de/insidedataassociation/data.html*

Table 4 The large simulation data sets and the algorithms implemented

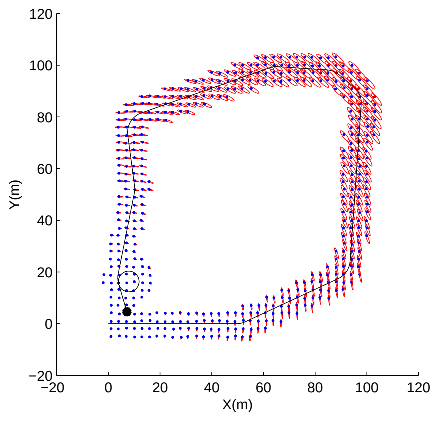
Data set	environment	observations	features	Algorithms implemented
3433 time steps	$120 \times 120m^2$	21677	461	EKF sequential, EKF batch, DMJ, I-DMJ, SJSJF, I-SLSJF, ML
8240 time steps	$150 \times 150m^2$	40524	612	EKF batch, DMJ, I-DMJ, SJSJF, I-SLSJF, ML
35187 time steps	$300 \times 300m^2$	219332	4202	I-DMJ, DMJ, I-SLSJF, SLSJF

Table 5 Consistency evaluations using NEES for different algorithms using the large simulation data sets: P — final estimate pass the χ^2 test, F - final estimate fail the χ^2 test, NI — algorithm not implemented, in the parentheses — total number of passes/total number of evaluations

Time steps	EKF sequential	EKF batch	DMJ	SLSJF	I-DMJ	I-SLSJF	ML
3433	F (3341/3433)	P (3433/3433)	P (100/100)	P (100/100)	P (100/100)	P (100/100)	P (3433/3433)
8240	NI	F (2757/8240)	F (45/50)	F (45/50)	P (50/50)	P (50/50)	P (8240/8240)
35187	NI	NI	F (305/700)	P (628/700)	P (700/700)	P (636/700)	NI



(a) Map obtained by EKF with sequential update



(b) The map obtained by EKF with batch update

Fig. 6 Simulation results for 3433 loops data – EKF with sequential update fails

correct data association is given in the data set and it is not performed here in this paper. In this data set, there are $p = 3298$ robot poses, $n = 576$ landmarks and $m = 14309$ measurements.

Fig. 9(a) shows the map generated by ML. The data is separated into 200 parts and each part of the data is used to build a small local map by ML. Fig. 9(b) shows the global map obtained by joining the 200 local maps using I-DMJ (189 admissible local maps were constructed from the 200 local maps using the process described in Section 3.3).

5.2.1 Consistency comparison

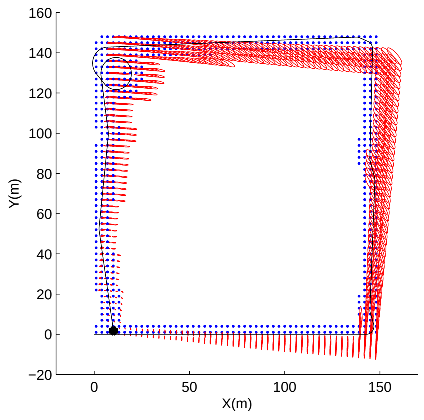
EKF and I-SLSJF are also implemented using this data set. By overlapping the EKF map, I-DMJ map and I-SLSJF map with the ML map, the differences between the estimated feature positions become clear (see Fig. 9(c) to Fig. 9(h)).

For the consistency comparison, we follow the idea presented in Section 4.1.5.

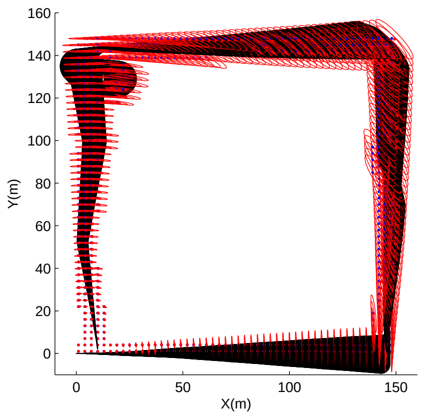
For this data set, the relative position among four features (basically in the 4 corners of the building, see Fig. 10) are measured by hand. The position is given in a coordinate system aligned with the building walls with the outdoor section on the right side. Hence the positions are not given in the SLAM coordinate system, which has the not precisely known starting pose as the origin.

Now the approach described in Case 1 in Section 4.1.5 is used to evaluate the consistency of the different algorithms. The first row of Table 6 shows the χ^2 test of the four algorithms. It can be seen that all the algorithms fail the test. This may mean that none of the mapping results is of good quality as compared with the ground truth.

We also did a consistency check using the ground truth data association following the process described in Case 2 in Section 4.1.5. For this data set, the largest loop is closed at step 2398 when feature with ID 174 is re-observed. To compare the consistency of the different algorithms, we deliberately change the observation made at step 2398 (the feature ID 174 is changed to



(a) Map obtained by EKF (red: 2σ covariance ellipses of the features estimate, blue: the true robot trajectory)



(b) The map obtained by ML

Fig. 7 Simulation results of 8240 loops data – EKF SLAM fails

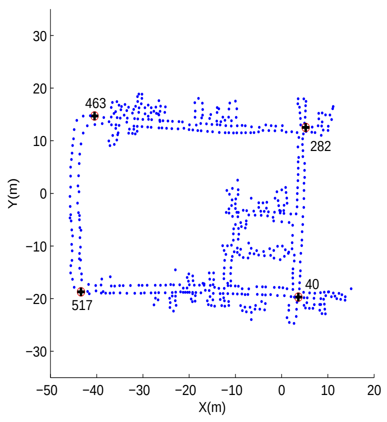


Fig. 10 Four features with relative coordinates hand measured in the DLR-Spatial-Cognition data set. Feature with ID 40: $(0 \pm 0, 0 \pm 0)$; Feature 282: $(-0.10 \pm 0.1, -34.27 \pm 0.1)$; Feature 463: $(48.68 \pm 0.1, -34.65 \pm 0.1)$; Feature 517: $(49.86 \pm 0.1, -0.05 \pm 0.1)$.

999) such that the algorithms will treat this feature as a new feature and the loop will not be closed. Using the data up to 2398 steps (with the last step data changed), the estimation results from different algorithms are obtained.

Fig. 11(a) to Fig. 11(d) show the maps and the 2σ covariance ellipses of the two features (actually they are the same feature). The second row of Table 6 shows the χ^2 tests of the null hypothesis that the two features are the same. It can be seen that the EKF result fails the test but the other three solutions appears to be reasonable.

This explains the loop closing χ^2 tests in Table 6 but not the large inconsistency with ground truth. Further investigations revealed as scaling error in feature distances [31, §6.2.5] which can indeed cause inconsistency with ground truth but not with loop-closure.

5.2.2 Accuracy comparison

For accuracy comparison, we use the approach as discussed in Section 4.2.2.

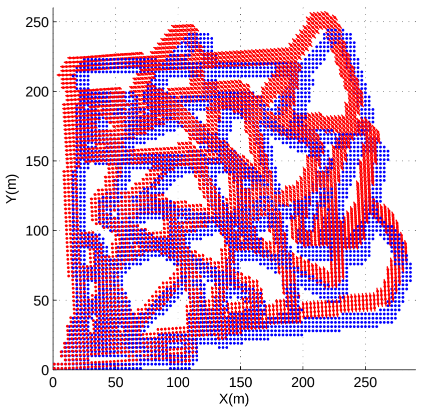
Table 7 shows the χ^2 errors from different algorithms. In theory, the exact minimum of the χ^2 error should be $2n_M - 2n_X = 27466$ (the dimension of the measurements minus the dimension of the state) on average. This indicates that the measurement covariances in the data set are overconfident by a factor of $56871/27466 = 2.07$. Table 7 also shows the error ratio relative to ML as defined by (30). It can be seen that the additional error introduced by I-DMJ is significantly larger than that of I-SLSJF and EKF.

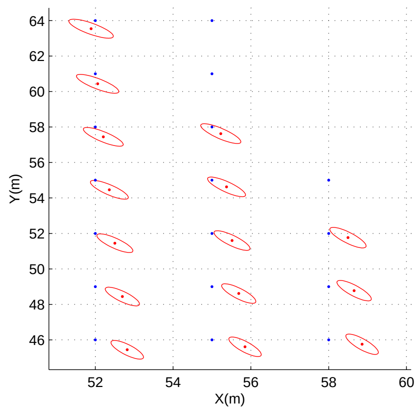
5.2.3 Efficiency comparison

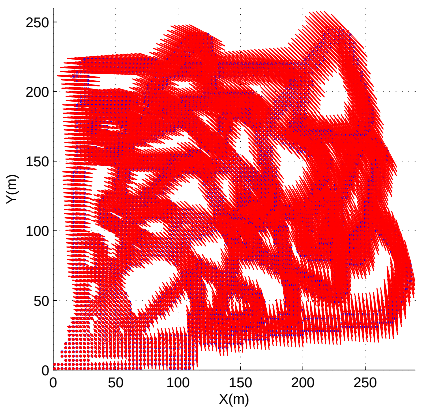
Table 8 compares the number of non-zero elements and the times of access to the non-zeros for each algorithms. For this data set, the number of non-zeros for I-DMJ (I-SLSJF) is around 1/6 (1/3) of the non-zeros for ML. The times of access to the non-zeros for I-DMJ (I-SLSJF) is around 1/70 (1/35) of that of ML.

5.3 Experimental results using Victoria Park data set

The I-DMJ, I-SLSJF, ML and EKF algorithms were also applied to the popular Victoria Park data set which was first used in [17]. Neither ground truth nor noise parameters are available for this data set. Published


(a) The map obtained by fusing 700 local maps by DMJ


(b) A close look of the map by DMJ: dots are true feature positions, 2σ covariance ellipses are from DMJ


(c) The map obtained by fusing 700 local maps by I-DMJ

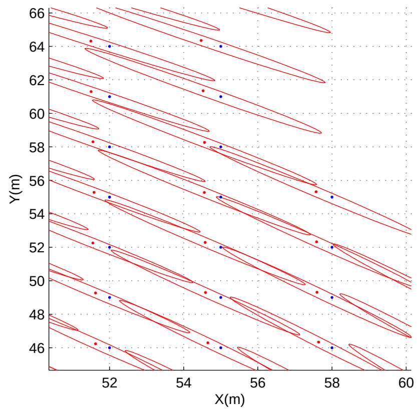

(d) A close look of the map by I-DMJ: dots are true feature positions, 2σ covariance ellipses are from I-DMJ

Fig. 8 Simulation results using 35187 step data set - map fusion without smoothing fails

Table 6 The χ^2 tests for consistency check — DLR-Spatial-Cognition data set

method	99% confidence gate	EKF (batch)	I-DMJ	I-SLSJF	ML
relative position among the four features	15.0863	1511.2	2379.0	1552.6	1557.5
true data association	9.2103	39.8163	6.6268	10.8553	9.9742

results for the vehicle trajectory and uncertainty estimates vary [1][9][16][17], presumably due to different parameters used by various researchers.

Due to the presence of "moving" features, different SLAM algorithms obtain slightly different data association results and it is impossible to be certain as to the correct data associations in this data set. In this paper, the odometry information and the observations made from the 6898 poses to 299 good quality features are used (the others are treated as outliers) and the same data association result is used in different algorithms in order to compare their performance [5].

Fig. 12(a) presents the vehicle poses and the feature position estimates using ML. Two hundred local maps are built by ML. Fig. 12(b) shows the global map obtained by joining the 200 local maps using I-DMJ.

[5] For Victoria Park data set, we have made the preprocessed sensor data and the parameters we used in this paper available on the website: http://services.eng.uts.edu.au/~sdhuang/research.htm. We hope this is useful for people to get a quick start in implementing their SLAM algorithm to this data set, although there is no guarantee that the data and parameters are 100% accurate.

Table 7 Non-linear χ^2 error of the different estimates. The χ^2 errors for EKF, I-DMJ, and I-SLSJF were obtained by running Gauss-Newton on the full non-linear SLAM problem while fixing the feature positions to the respective estimate.

	data set	ML	EKF	I-DMJ	I-SLSJF
$\min_{X_r} \chi^2 \begin{pmatrix} \hat{X}_f \\ X_r \end{pmatrix}$	DLR-Spatial-Cognition	56871	57159	63839	56898
error ratio relative to ML defined by (30)	DLR-Spatial-Cognition	0	0.25	6.05	0.02
$\min_{X_r} \chi^2 \begin{pmatrix} \hat{X}_f \\ X_r \end{pmatrix}$	Victoria Park	45008	45184	45063	45041
error ratio relative to ML defined by (30)	Victoria Park	0	0.29	0.09	0.06

5.3.1 Consistency comparison

EKF and I-SLSJF are also implemented using this data set. The EKF map, I-DMJ map, and I-SLSJF map are overlapped with the ML map in Fig. 12(c), Fig. 12(e), and Fig. 12(g), respectively. It can be seen that the estimates of different algorithms are almost identical. Close look at the maps show that the covariance ellipses obtained by EKF, I-DMJ and I-SLSJF are slightly smaller than that from ML, meaning that the three algorithms are slightly over confident on their estimates.

5.3.2 Accuracy comparison

The second row of Table 7 shows the χ^2 errors and error ratios relative to ML from different algorithms. Their difference from the ML χ^2 error are all very small. This may be due to the clean sensor data used and the accuracy of the laser sensors.

5.3.3 Efficiency comparison

Table 8 compares the size of the state vector, the number of non-zeros in the sparse information matrix, and the total times the non-zeros have been accessed. It is clear that the computational cost of I-DMJ and I-SLSJF are significantly less than that of EKF and ML.

6 Conclusion

This paper first introduced a new local map joining algorithm – I-DMJ, then the metrics for comparing the consistency, accuracy and efficiency of different mapping algorithms are proposed. These metrics are applied to compare the I-DMJ algorithm with some existing SLAM algorithms.

Simulation results show that both the generalized eigenvalue approach and the NEES approach are capable of providing a good indication of the consistency of different algorithms. However, comparing the consistency of algorithms using experimental data is extremely difficult. Only some necessary conditions for algorithm consistency can be evaluated unless the ground truths of data association, feature positions and robot trajectory are all available. The total times the non-zero elements in the information matrices are accessed appears to be a good metric for computational cost.

In comparison to many other mapping algorithms, it appears that I-DMJ provides a good balance in the tradeoffs among consistency, accuracy and efficiency. In most of the simulation and experimental scenarios, I-DMJ can efficiently produce estimates with acceptable consistency and accuracy.

Many improvements can be made to the I-DMJ algorithm to reduce the computational cost further, for example, by exploiting the idea of divide and conquer [15] [23] or treemap representation [21] for the map joining instead of the sequential map joining approach used now. However, how to solve the data association problem when using these strategies is a research issue. That needs to be addressed in the future.

It should be noted that this paper focuses only on point feature based SLAM algorithms. Also, it is mainly concerned with theoretical evaluation of different algorithms instead of their practical applications. Further work is necessary to analyze the performance metrics of the SLAM algorithms in real applications using more experimental data.

Acknowledgements 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. This work has also been supported by DFG grant SFB/TR 8 Spatial Cognition.

References

1. 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.
2. T. Bailey and H. Durrant-Whyte, "Simultanouse localization and mapping (SLAM): Part II". *IEEE Robotics & Automation Magazine*, vol. 13, Issue 3, Sept. 2006, pp. 108-117.
3. U. Frese. "A discussion of simultaneous localization and mapping". *Autonomous Robots*, Volume 20, Issue 1, (January 2006), Pages: 25-42.
4. T. Bailey, J. Nieto, J. Guivant, M. Stevens and E. Nebot. "Consistency of the EKF-SLAM algorithm". In *Proceedings of*

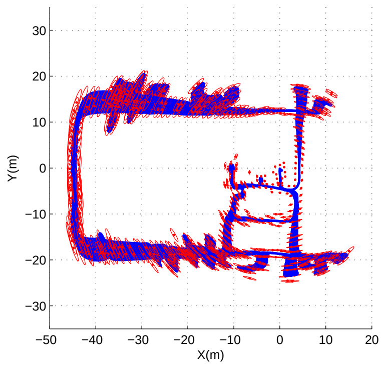
Table 8 Compare the dimension of the state vector, the number of non-zero elements in the final information matrix, and the total number of access to the non-zero elements in the information matrix (covariance matrix)

Data set	Algorithm	dim of state	(num of poses, num of features)	num of non-zeros	num of access
Simulation 3433 steps	EKF Sequential	925	(1 pose, 461 features)	855625	7618648976
	EKF Batch	925	(1 pose, 461 features)	855625	1232733824
	ML	11221	(3433 poses, 461 features)	354557	4989335096
	I-SLSJF	1222	(100 poses, 461 features)	53786	16295082
	SLSJF	1222	(100 poses, 461 features)	53786	5729047
	I-DMJ	922	(0 pose, 461 features)	32964	10433606
	DMJ	922	(0 pose, 461 features)	32964	4008856
Simulation 8240 steps	EKF (Batch)	1227	(1 pose, 612 features)	1505529	2798910271
	ML	25944	(8240 poses, 612 features)	711150	78220927724
	I-SLSJF	1374	(50 poses, 612 features)	88032	11089000
	SLSJF	1374	(50 poses, 612 features)	88032	3170896
	I-DMJ	1224	(0 pose, 612 features)	68936	6513798
	DMJ	1224	(0 pose, 612 features)	68936	2363358
Simulation 35187 steps	I-SLSJF	10504	(700 poses, 4202 features)	583798	1418540749
	SLSJF	10504	(700 poses, 4202 features)	583798	510892091
	I-DMJ	8404	(0 pose, 4202 features)	387472	1607152858
	DMJ	8404	(0 pose, 4202 features)	387472	971918144
Victoria Park	EKF (Batch)	601	(1 pose, 299 features)	361201	1202307021
	ML	21292	(6898 poses, 299 features)	732704	8319832357
	I-SLSJF	1198	(200 poses, 299 features)	137416	105919488
	I-DMJ	598	(0 pose, 299 features)	62604	68685002
DLR-Spatial-Cognition	EKF (Batch)	1155	(1 pose, 576 features)	1334025	1691188295
	ML	11043	(3297 poses, 576 features)	263013	3330175088
	I-SLSJF	1752	(200 poses, 576 features)	86881	89455649
	I-DMJ	1152	(0 pose, 576 features)	42368	47206954

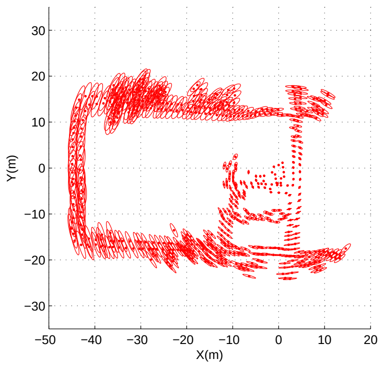
the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3562-3568, Beijing, China, October 9-15, 2006.

5. 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.
6. 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.
7. G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of Extended Kalman Filter-based SLAM", In *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, CA, May 19-23 2008, pp. 473-479.
8. 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.
9. 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.
10. S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.
11. 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.
12. 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/
13. 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.
14. T. P. Speed, and H. T. Kiiveri, "Gaussian Markov distributions over finite graphs". *The Annals of Statistics*, 14(1): 138-150, 1986.
15. 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.
16. 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.
17. 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.
18. 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.
19. J. Folkesson and H. I. Christensen, "Closing the loop with Graphical SLAM", *IEEE Transactions on Robotics*, 2007, vol. 23, no. 4, pp. 731-741.
20. 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.

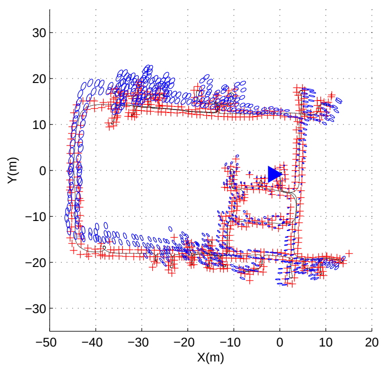
21. U. Frese, “Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping”. *Autonomous Robots*, 21(2): 103-122, 2006.
22. P. Krauthausen, F. Dellaert, and A. Kipp, “Exploiting locality by nested dissection for square root smoothing and mapping”, *In Proceeding of Robotics: Science and Systems*, Philadelphia, U.S.A. 2006.
23. 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 Robotics: Science and Systems*, June 27-30, Atlanta, USA.
24. U. Frese and 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.
25. 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.
26. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes, Second Edition.* Cambridge University Press, Cambridge, 1992.
27. S. Huang, Z. Wang and G. Dissanayake. “Sparse local submap joining filter for building large-scale maps”. *IEEE Transactions on Robotics*, 2008, Vol. 24, No. 5, 1121-1130, October 2008.
28. L. M. Paz, P. Pinies, J. Neira, and J. D. Tardos. “Global localization in SLAM in bilinear time”. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 655-661, Edmonton, Alberta, Canada, August 2-6, 2005.
29. S. Huang, Z. Wang, G. Dissanayake, and U. Frese, “Iterated SLSJF: A sparse local submap joining algorithm with improved consistency”, *2008 Australiasan Conference on Robotics and Automation*. Canberra, December 2008. Available online: http://www.araa.asn.au/acra/acra2008/papers/pap102s1.pdf
30. G. Grisetti, D. L. Rizzini, C. Stachniss, E. Olson and W. Burgard, “Online constraint network optimization for efficient maximum likelihood mapping”. *In Proceedings of 2008 IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, California, on May 19-23, 2008.
31. C. Hertzberg, *A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds*, Master Thesis, University of Bremen, 2008.



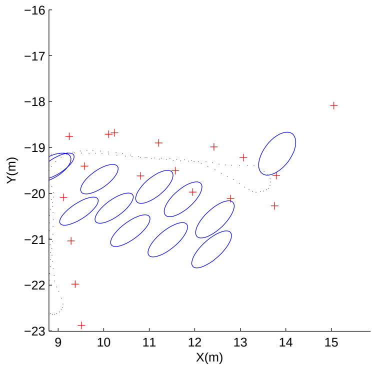
(a) Map obtained by ML



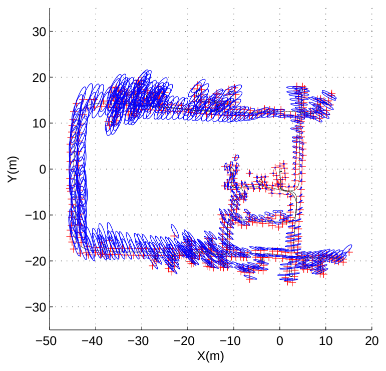
(b) Map obtained by joining 200 local maps using I-DMJ



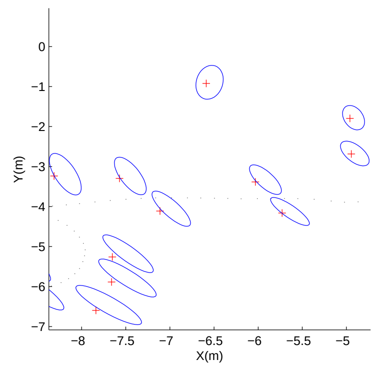
(c) Overlap of EKF map with ML map



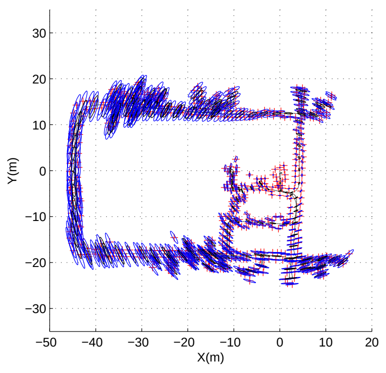
(d) Overlap of EKF map with ML map – close look



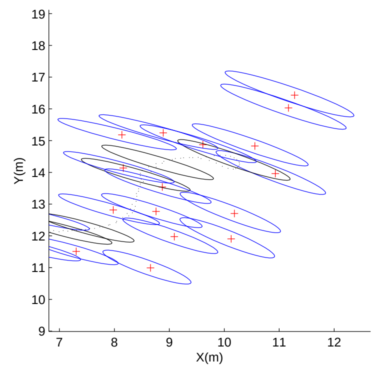
(e) Overlap of I-DMJ map with ML map



(f) Overlap of I-DMJ map with ML map – close look

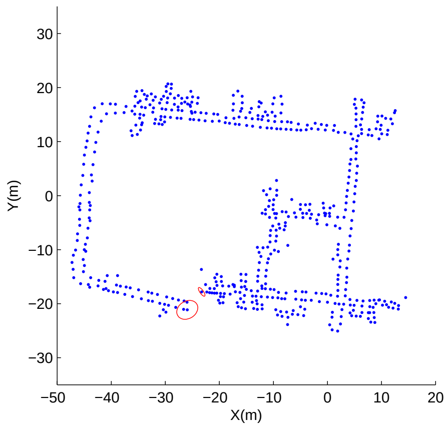


(g) Overlap of I-SLSJF map with ML map

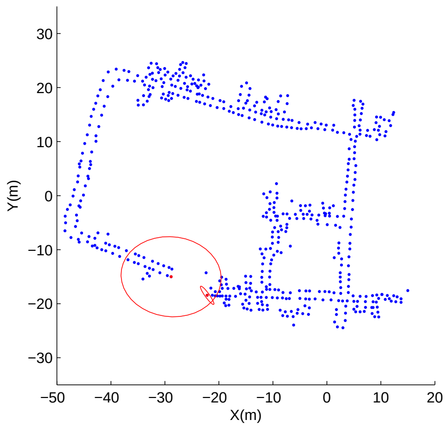


(h) Overlap of I-SLSJF map with ML map – close look

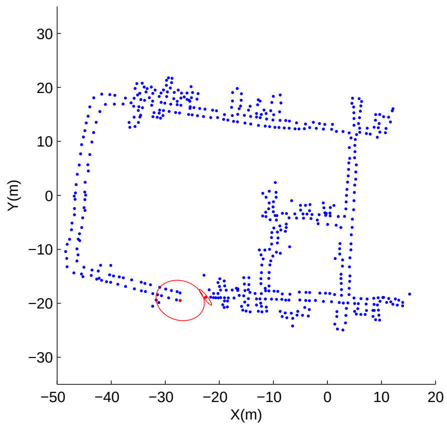
Fig. 9 Compare the maps from different algorithms using DLR-Spatial-Cognition data set. In Figures 9(c) to 9(h), the (red) crosses are the estimated feature positions from ML, the (blue) ellipses are the 2σ covariance ellipses from the other algorithm.



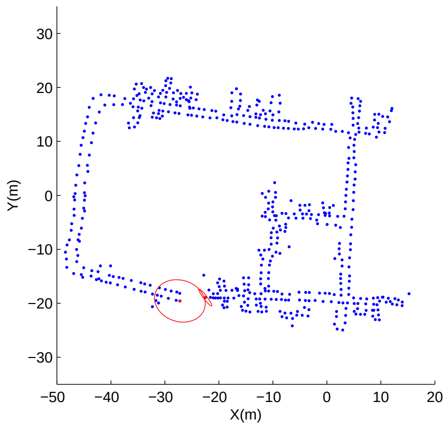
(a) EKF SLAM result



(b) Map obtained by joining 200 local maps using I-DMJ (191 "admissible" local maps were built from the 200 local maps)

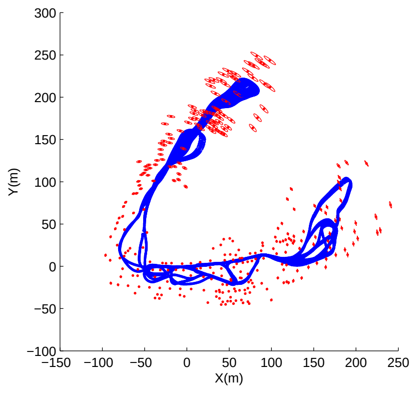


(c) Map obtained by joining 200 local maps using I-SLSJF

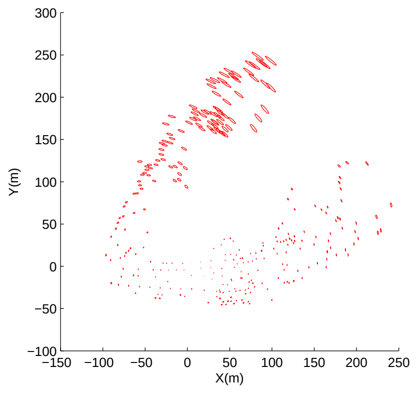


(d) Map obtained by ML

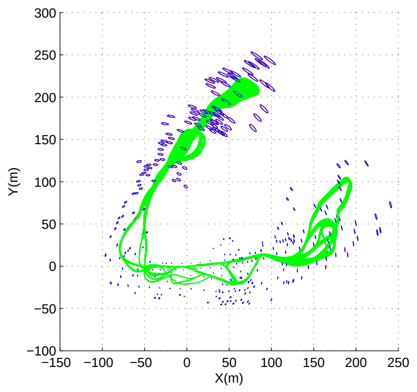
Fig. 11 The results using the first 2938 step data of DLR-Spatial-Cognition data set. The (blue) dots are the estimated positions of all the features. The two features with (red) 2σ covariance ellipses around them are actually the same feature but we deliberately allocate different IDs to them such that all the algorithms treat them as different features. The true data association results of all the other features is used in the different algorithms. It can be seen, that except for EKF the result is roughly consistent.



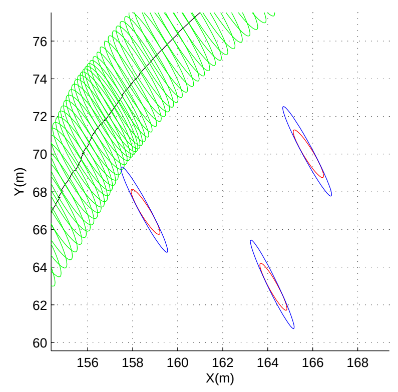
(a) ML result



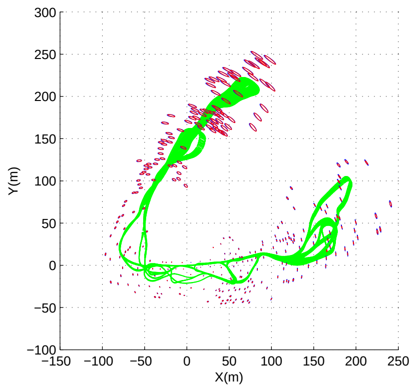
(b) Map by joining 200 local submaps using I-DMJ



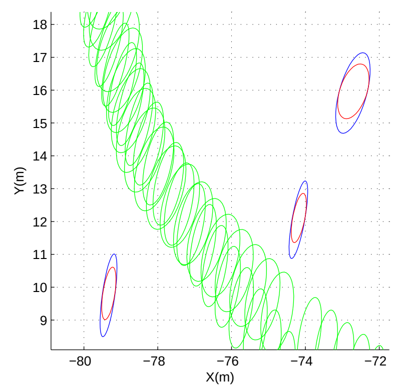
(c) Overlap of EKF map with ML map



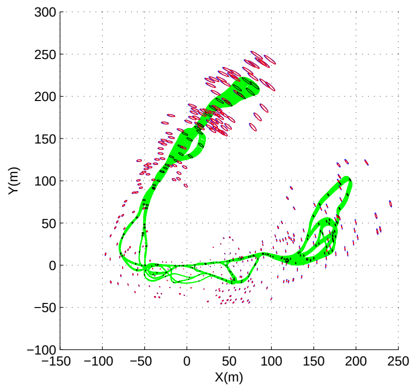
(d) Overlap of EKF map with ML map – close look



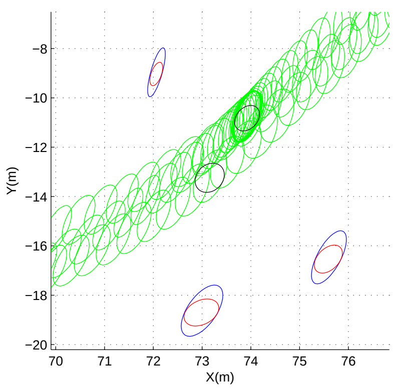
(e) Overlap of I-DMJ map with ML map



(f) Overlap of I-DMJ map with ML map – close look



(g) Overlap of I-SLSJF map with ML map



(h) Overlap of I-SLSJF map with ML map – close look

Fig. 12 Comparing the maps obtained by different algorithms – Victoria Park data set. In Figures 12(c) to 12(h), the larger ellipses (blue) for features are the 2σ covariance ellipses obtained from ML.