D-SLAM: A Decoupled Solution to Simultaneous Localization and Mapping

Zhan Wang Shoudong Huang Gamini Dissanayake
ARC Centre of Excellence for Autonomous Systems (CAS)
Faculty of Engineering, University of Technology, Sydney, Australia
{zwang, sdhuang, gdissa}@eng.uts.edu.au
http://www.cas.edu.au

Abstract

The main contribution of this paper is the reformulation of the simultaneous localization and mapping (SLAM) problem for mobile robots such that the mapping and localization can be treated as two concurrent yet separated processes: D-SLAM (decoupled SLAM). It is shown that SLAM with a range and bearing sensor in an environment populated with point features can be decoupled into solving a nonlinear static estimation problem for mapping and a low-dimensional dynamic estimation problem for localization. This is achieved by transforming the measurement vector into two parts: one containing information relating features in the map and another with information relating the map and robot. It is shown that the new formulation results in an exactly sparse information matrix for mapping when it is solved using an Extended Information Filter (EIF). Thus a significant saving in the computational effort can be achieved for large-scale problems by exploiting the special properties of sparse matrices. An important feature of D-SLAM is that the correlation among features in the map are still kept and it is demonstrated that the uncertainty of the feature estimates monotonically decreases. The algorithm is illustrated and evaluated through computer simulations and experiments.

Keywords: Decoupled SLAM, Extended Information Filter, Sparse Matrix, Computational Complexity

1 Introduction

The process of building a map of an environment while concurrently generating an estimate for the location of an autonomous vehicle is known as simultaneous localization and mapping (SLAM). The SLAM problem has been the subject of extensive research in the past few years with a number of robotics research groups contributing to make substantial progress in this area (see for example [8], [14], [30] and the references therein). In these publications, among many others, the fact that the estimation of the robot location and the map needs to be simultaneous has been well demonstrated and issues ranging from proofs of convergence to computational efficiency have been addressed.

The structure of the estimation-theoretic formulation of SLAM makes it possible to observe a feature in the map and use the information present in this observation to improve the estimation of the location of all features in the map, whether these features are in the current neighborhood or very far way. This is quite a powerful effect in the sense that when the robot revisits parts of the map that are more accurately known, strong corrections to the location estimates of the features in less well defined regions of the map occur, leading to a perfect map in the limit. In traditional SLAM, this effect comes from maintaining the correlations among the estimates of the robot location and all the feature locations. It is well known that maintaining a state vector consisting of locations of the robot and all the features and the associated full covariance matrix leads to a heavy computational burden when solving large-scale SLAM problems.

In this paper, a decoupled solution to the SLAM problem (D-SLAM) is provided. It is demonstrated that the SLAM problem can be reformulated such that the state vector for mapping only contains the locations of all the features while the state vector for localization only contains the locations of the robot and local features. It is shown that although localization and mapping processes are decoupled, the correlations among the feature location estimates are still maintained, and when formulated in the information form, the estimation problem can be solved at a cost of $O(N)$, where N is the total number of features in the environment.

1.1 Related work

A variety of attempts have been made to remove the robot location from the state vector in order to reduce the computational complexity of SLAM. For example, Newman [24] introduced a relative map in which the map state contains the relative locations among the features. Csorba et al. [5], Deans and Herbert [6], Pradalier and Sekhavat [25] and Martinelli [19] have made use of relative maps where the map state contains relative distances and/or angles among the features, which are invariants under shift and rotation. The structure of the covariance matrix is kept sparse by maintaining a state vector with redundant elements. As the relationships between the map elements are not enforced, for large-scale problems the map becomes complex and difficult to use. However, if the constraints that enforce these relationships are applied, the simple structure of the covariance matrix is destroyed, leading to an increased computational complexity.

Use of an Extended Information Filter (EIF) to solve SLAM in order to gain a computational saving has also been demonstrated by a number of researchers. A notable result has been that from Thrun et al. [30] where a sparsification process is used to reduce the number of non-zero elements in the information matrix resulting in significant computational savings. Although Frese [13] provided a proof for the approximate sparseness of the information matrix, Eustice et al. [10] demonstrated that the process of sparsification of the information matrix leads to inconsistent estimates.

In a recent development, Eustice et al. [9] show that the inclusion of the robot trajectory in the form of past robot poses in the state vector leads to an exactly sparse information matrix. The resulting Exactly Sparse Delayed State Filter (ESDSF) provides clear computational advantages when a view-based map representation is used, where a sequence of robot poses are used to describe the map. In the example presented in [9], the map is not represented within the state vector and is therefore not directly updated. When both features and robot poses are included in the state vector, the EIF also provides an exactly sparse information matrix [7], [15]. However, in this scenario the sparseness of the information matrix is achieved through increasing the state dimension. The state dimension keeps increasing even when robot is revisiting previous explored regions. This limitation has been overcome by Walter et al. [31] who achieve an exactly sparse information matrix with just the last robot pose included in the state vector. The information matrix is controlled to be sparse by deliberately "kidnapping" and "relocating" the robot from time to time. Further comments on the relationship between this work and D-SLAM is presented in Section 7.

1.2 Contributions

This paper provides a novel decoupled solution to SLAM, where the state vector for mapping contains the absolute locations of the features ($2N$ dimension for N features in a 2D environment). The main contributions of this paper are the following.

- It is shown that the decoupling of mapping and localization is possible using an absolute map with no redundant elements in the state vector. This is achieved by recasting the range and bearing observation into a new equivalent observation containing information about the relative locations among the features.

- It is demonstrated that the new formulation naturally results in an exactly sparse information matrix for mapping. Approximating near zero elements is not required to achieve sparseness. Furthermore, it is shown that the information matrix retains its sparseness even after loop closures and that the extent of sparseness is related to the range of the sensor on board the robot and feature density in the environment.

- Using a new iterative preconditioning algorithm to reduce the computational cost of state vector and (part of) the covariance matrix recovery, it is demonstrated that the overall computational cost of D-SLAM algorithm is of $O(N)$ where N is the total number of features in the environment.

The paper is organized as follows. In Section 2, the process of recasting the observations and the key idea of D-SLAM are stated. The details of the mapping and localization processes in the D-SLAM algorithm are provided in Section 3. Section 4 addresses some implementation issues in D-SLAM including data association, state recovery and admissible measurements. The computational complexity is analyzed in Section 5. Experimental and simulation results are presented in Section 6 to evaluate the algorithm. Section 7 concludes the paper and addresses future research directions.

2 Extracting map information from observations

Observations made from a sensor mounted on the robot contain the relative location of the features with respect to the robot. Using these observations directly in SLAM makes the robot and feature location estimates correlated. In order to decouple mapping and localization in SLAM, a key step is to extract map information from the observations. This can be achieved by transforming the measurement vector into one consisting of two parts: one part containing distances and angles among features; the other relating the features and the robot location. It is important to formulate the information extraction process in order to (1) minimize information loss to maintain efficiency, and (2) avoid information reuse.

To maintain notational simplicity and improve clarity, it is assumed that all observations contain measurements to at least two features that are previously seen and already present in the map. A strategy to overcome this limitation by combining a number of measurements into one admissible measurement is described in Section 4.3.

2.1 Partitioning the measurement vector

Suppose the robot observes m features $f_1, \cdots, f_m$ at a particular time, among which f_1 and f_2 have been previously seen.

2.1.1 The original measurements

The original measurement vector contains the measured range and bearing of each observed feature:

$$z_{old} = \left[\ r_1, \theta_1, \cdots, r_m, \theta_m \ \right]^T, \tag{1}$$

which contains noise, assumed to be Gaussian with zero mean and covariance matrix

$$R_{old} = diag \left[\ \sigma_{r_1}^2, \sigma_{\theta_1}^2, \cdots, \sigma_{r_m}^2, \sigma_{\theta_m}^2 \ \right]. \tag{2}$$

2.1.2 Distances and angles w.r.t. f_1 and f_2

The measurement vector can be transformed to z_{new} written as

$$\begin{bmatrix} \alpha_{r12} \\ d_{1r} \\ \alpha_{\phi 12} \\ --- \\ d_{12} \\ \alpha_{312} \\ d_{13} \\ \vdots \\ \alpha_{m12} \\ d_{1m} \end{bmatrix} = \begin{bmatrix} atan2\left(\frac{-\tilde{y}_1}{-\tilde{x}_1}\right) - atan2\left(\frac{\tilde{y}_2-\tilde{y}_1}{\tilde{x}_2-\tilde{x}_1}\right) \\ \sqrt{(-\tilde{x}_1)^2 + (-\tilde{y}_1)^2} \\ -atan2\left(\frac{\tilde{y}_2-\tilde{y}_1}{\tilde{x}_2-\tilde{x}_1}\right) \\ --- \\ \sqrt{(\tilde{x}_2 - \tilde{x}_1)^2 + (\tilde{y}_2 - \tilde{y}_1)^2} \\ atan2\left(\frac{\tilde{y}_3-\tilde{y}_1}{\tilde{x}_3-\tilde{x}_1}\right) - atan2\left(\frac{\tilde{y}_2-\tilde{y}_1}{\tilde{x}_2-\tilde{x}_1}\right) \\ \sqrt{(\tilde{x}_3 - \tilde{x}_1)^2 + (\tilde{y}_3 - \tilde{y}_1)^2} \\ \vdots \\ atan2\left(\frac{\tilde{y}_m-\tilde{y}_1}{\tilde{x}_m-\tilde{x}_1}\right) - atan2\left(\frac{\tilde{y}_2-\tilde{y}_1}{\tilde{x}_2-\tilde{x}_1}\right) \\ \sqrt{(\tilde{x}_m - \tilde{x}_1)^2 + (\tilde{y}_m - \tilde{y}_1)^2} \end{bmatrix} \tag{3}$$

where

$$\tilde{x}_i = r_i \cos\theta_i, \quad \tilde{y}_i = r_i \sin\theta_i, \quad i = 1, \cdots, m. \tag{4}$$

The physical meaning of the new measurement vector is shown in Figure 1(b) while that of the original measurement vector shown in Figure 1(a).

The last $2m-3$ elements in the measurement vector shown in (3) contain information about distances and angles among features that are independent of the coordinate system. The first three elements depend on the robot pose and features f_1, f_2. This part carries information about the robot location. Thus the measurement vector can be naturally partitioned into two vectors denoted as

$$z_{rob} = \begin{bmatrix} \alpha_{r12} \\ d_{1r} \\ \alpha_{\phi 12} \end{bmatrix}, \quad z_{map} = \begin{bmatrix} d_{12} \\ \alpha_{312} \\ d_{13} \\ \vdots \\ \alpha_{m12} \\ d_{1m} \end{bmatrix}. \tag{5}$$

There is a one to one correspondence between z_{old} and $z_{new} = \left[z_{rob}^T, z_{map}^T\right]^T$. Furthermore, z_{new} is constructed such that it contains the minimum number of elements required to completely capture the total information content contained in z_{old}. It is important, however, to note that the two measurement vectors z_{rob} and z_{map} are not independent. Thus the proposed transform does not completely divide the information contained in the original measurements into two parts. Therefore the estimation process that exploits these new measurement vectors needs to be structured appropriately in order to avoid statistical inconsistency.

2.1.3 Measurement noise covariances

For a Gaussian random (vector) variable x with a mean $\bar{x}$ and a covariance matrix R_x, any (vector) function of x, $g(x)$, can be approximated by a Gaussian provided x is near $\bar{x}$. Mean of this Gaussian is $g(\bar{x})$ and its covariance matrix is $\nabla g R_x \nabla g^T$ where ∇g is the Jacobian of g with respect to x evaluated at $\bar{x}$. This relationship can be used to compute the covariance matrix of the noise on the new measurement vector z_{map} from (2), (3), (4) and (5).

2.2 The key idea of D-SLAM

In D-SLAM, the key idea is to use z_{map} to estimate a state vector containing only the locations of features in the environment. As shown in the following sections, the absence of the robot location from the state vector results in an estimator structure that offers significant computational advantages.

As z_{rob} and z_{map} are not independent, z_{rob} also contains some information about the map which is not exploited during the mapping process, resulting in some information loss. Furthermore, this dependency makes it important that the localization process is formulated carefully in order to avoid information reuse. The details of the mapping and localization algorithms in D-SLAM are described in the next section.

3 Decoupled estimators for localization and mapping

3.1 Mapping in D-SLAM

3.1.1 State vector

The state vector used for mapping only contains the locations of the features:

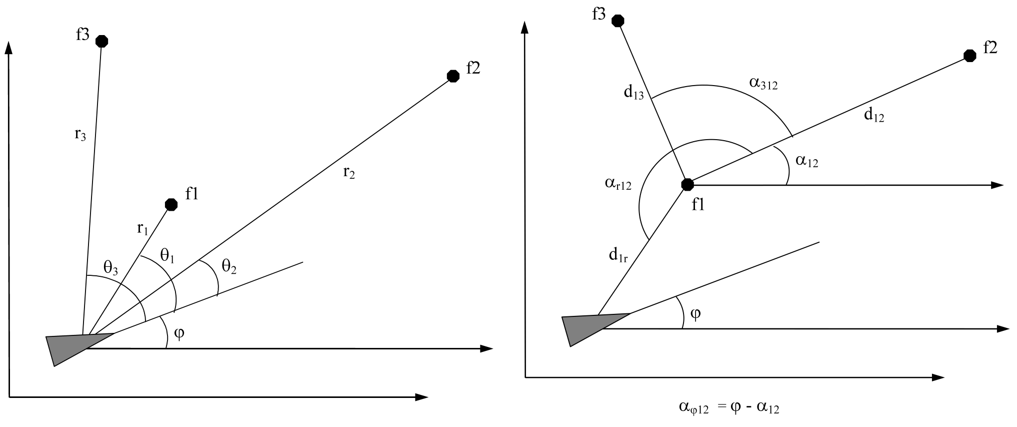
$$X = (X_1, \cdots, X_n)^T = (x_1, y_1, x_2, y_2, \cdots, x_n, y_n)^T, \tag{6}$$

where $X_1, \cdots, X_n$ are the states of features $f_1, \cdots, f_n$.

For convenience, the initial robot pose is used to define the coordinate system, where the origin is the robot position and the x-axis is along the initial robot heading. The information vector $i(k)$ is defined as

$$i(k) = I(k)\hat{X}(k) \tag{7}$$

where $I(k)$ is the information matrix which is the inverse of the state covariance matrix $P(k)$ and $\hat{X}(k)$ is the estimate of the state. As the features are stationary, mapping reduces to a static non-linear estimation problem which can be efficiently formulated in



(a) Original range and bearing measurements.

(b) Geometric interpretation of the measurement vector given by equation (3) — $d_{12}, \alpha_{312}, d_{13}$ are distances and angles among features, which contain information about the map only, and are used in D-SLAM mapping.

Figure 1: Relationship between the original measurement vector and the recast measurement vector

the information form using an Extended Information Filter (e.g. [20], [30]).

3.1.2 Measurement model

Suppose the robot observes m features $f_1, \cdots, f_m$, where two of these features f_1, f_2 have been previously seen. The recast measurement used for mapping is

$$\begin{aligned} z_{map} &= [d_{12}, \alpha_{312}, d_{13}, \cdots, \alpha_{m12}, d_{1m}]^T \\ &= H_{map}(X) + w_{map} \end{aligned} \tag{8}$$

where

$$H_{map}(X) = \begin{pmatrix} \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ atan2\left(\frac{y_3 - y_1}{x_3 - x_1}\right) - atan2\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \\ \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \\ \cdots \\ atan2\left(\frac{y_m - y_1}{x_m - x_1}\right) - atan2\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \\ \sqrt{(x_m - x_1)^2 + (y_m - y_1)^2} \end{pmatrix} \tag{9}$$

and w_{map} is the measurement noise whose covariance matrix R_{map} can be computed by (2), (3), (4) and (5).

3.1.3 Feature initialization and map update

Current location estimates of the previously seen features can be used together with d_{1i}, α_{i12} in z_{map} to compute the initial location of a new feature, f_i, as follows:

$$\begin{aligned} \alpha_{12} &= atan2(\tfrac{\hat{y}_2 - \hat{y}_1}{\hat{x}_2 - \hat{x}_1}) \\ \hat{x}_i &= \hat{x}_1 + d_{1i}\cos(\alpha_{12} + \alpha_{i12}) \\ \hat{y}_i &= \hat{y}_1 + d_{1i}\sin(\alpha_{12} + \alpha_{i12}). \end{aligned} \tag{10}$$

As part of the initialization process, the dimension of the information vector and the information matrix are also increased by adding an appropriate number of zeros.

The information vector and the information matrix can now be updated using the measurement z_{map} as follows:

$$\begin{aligned} I(k+1) &= I(k) + \nabla H_{map}^T R_{map}^{-1} \nabla H_{map} \\ i(k+1) &= i(k) + \nabla H_{map}^T R_{map}^{-1} [z_{map}(k+1) - \\ &\quad H_{map}(\hat{X}(k)) + \nabla H_{map} \hat{X}(k)] \end{aligned} \tag{11}$$

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

3.1.4 Exactly sparse information matrix

In the matrix $\nabla H_{map}^T R_{map}^{-1} \nabla H_{map}$ in (11), all the elements relating to features that are not present in the measurement vector are exactly zero. This can be easily seen by the fact that

$$\nabla H_{map} = \left[\frac{\partial H_{map}}{\partial X_1}, \cdots, \frac{\partial H_{map}}{\partial X_m}, 0, \cdots, 0 \right]. \tag{12}$$

In a typical sensor where the sensor range is limited, only the features that are in the close proximity will be simultaneously observed. Therefore, if feature i and feature j are far away from each other, the measurement z_{map} will never contain both f_i and f_j. As the information matrix update involves a simple addition, the off-diagonal elements relating to i and j in the information matrix will remain exactly zero. Therefore, for a large-scale map, a significant portion of the information matrix will be exactly zero, resulting in an exactly sparse information matrix.

In general, each column (row) of the information matrix will contain at most a constant number of non-zero elements (this constant depends on the density of features and the sensor range), independent of the size of the environment and/or the total number of features.

3.2 Localization in D-SLAM

Given the current map, the robot location can be easily obtained by solving a "kidnapped robot problem" using the current observation. This, however, discards all knowledge of the previous robot location.

On the other hand, localization can also be done efficiently by an EKF-based local SLAM process where only the features in the vicinity of the robot are retained in the state vector. Features are removed from the state vector once they are not visible from the robot. This is effectively SLAM-aided dead reckoning which provides much better robot location estimate than that obtained using dead reckoning alone. However, the localization results will be far inferior to that obtained from a complete SLAM process as information contained in the features removed from the state vector are unrecoverable and thus the robot location estimate will not improve during loop closures.

In this work, it is proposed to combine the location estimates from these two approaches to achieve an improved estimate of robot location. The following subsections describe the localization process in more detail. Figure 2 shows a flow chart illustrating the localization process in D-SLAM.

3.2.1 State vector

At steps A and B of the flow chart in Figure 2, the state vector contains the robot and all the features observed at time k. Suppose the robot observes features $f_1, \cdots, f_m$ at time $k+1$, among which $f_1, \cdots, f_{m_1}, m_1 \leq m$ are features that have been previously seen and are already included in Map(k) (step F). The state vector at steps C, G and D contains the robot and those features observed at time $k+1$ which are in Map(k), $(X_r(k+1), X_1, \cdots, X_{m_1})$. The state vector at step E contains the robot and all the features observed at time $k+1$, $(X_r(k+1), X_1, \cdots, X_m)$.

3.2.2 Robot location estimate 1: SLAM with local features

In Figure 2, A to B is the prediction step based on the robot process model. Between B and C, the features that are not observed at time $k+1$ are deleted, previously deleted features that are reobserved at time $k+1$ are initialized and the state vector is updated. From D to E, the new features observed at time $k+1$ are initialized. Traditional EKF SLAM equations are used in these steps.

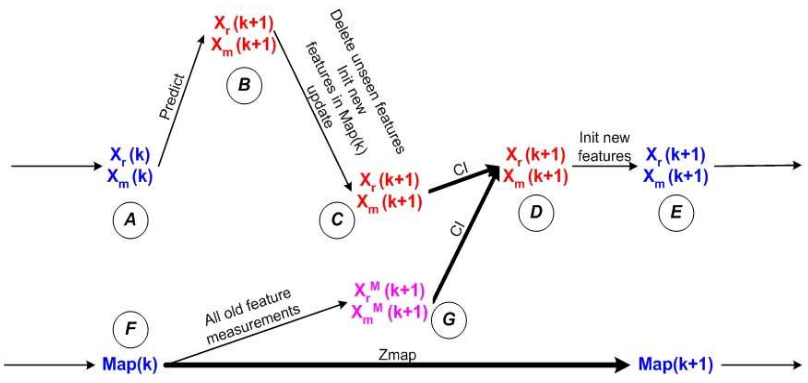


Figure 2: Flow chart of the localization and the mapping processes in D-SLAM

3.2.3 Robot location estimate 2: solution to the kidnapped robot problem

The process from F to G in Figure 2 is to solve the kidnapped robot problem which can be formulated in information form and solved as a linearized minimum mean square error estimation problem. The observations to features $f_1, \cdots, f_{m_1}, m_1 \leq m$ that have been previously seen are given by

$$\begin{aligned} z_{loc} &= (r_1, \theta_1, \cdots, r_{m_1}, \theta_{m_1})^T \\ &= H_{loc}(X_r(k+1), X_1, \cdots, X_{m_1}) + w_{loc}, \end{aligned} \tag{13}$$

where w_{loc} is the corresponding measurement noise. Using a process similar to that presented in Section 3.1.3, the location of the robot $X_r(k+1)$ together with an update for the rest of the state vector $X_1, \cdots, X_{m_1}$ can be computed.

3.2.4 Fusing the two robot location estimates

The local SLAM estimate is optimal, until the robot closes a loop by revisiting a previously traversed region of the map. The kidnapped robot solution will be superior when loop closures are present. These two estimates make use of the information from the same measurements, and thus are correlated. Maintaining the correlations is possible but requires significant computational effort. Covariance intersection (CI) (see [4] and [18]), that facilitates combining two correlated pieces of information, when the extent of correlation itself is unknown is, therefore, used to fuse these two estimates. The criteria used in computing the CI solution is selected to be minimizing the trace of the submatrix related to the robot pose in the covariance matrix. This process is illustrated by the links between C, G and D in Figure 2. The state vectors at steps C and G should contain the same features in the same order. Thus the state vector needs to be reordered between B and C. It is important to note that the kidnapped robot solution carries information from the mapping component of D-SLAM, and that this information will flow through to the estimate of the state vector used by the local SLAM. Thus the errors in the robot location estimate from the local SLAM process will stay bounded.

3.3 Consistency of D-SLAM

The information matrix in D-SLAM is exactly sparse. Therefore, further sparsification that have been shown to lead to inconsistencies [10], is not required to efficiently solve the mapping problem. Using CI to combine the two robot location estimates avoids possible inconsistency due to information reuse. Although the robot location computed by the process described in Section 3.2.4 is correlated to the map, this correlation does not affect the mapping process as the information about the robot location is not exploited during mapping. Thus the D-SLAM algorithm does not contain any approximations that can lead to estimator inconsistency. As the case with all EKF/EIF based estimation algorithms, however, it is possible that inconsistencies can occur during D-SLAM due to errors introduced by the linearization process.

4 Implementation issues

4.1 Data association

In the context of SLAM, data association refers to the process of associating the observations to the features in the map. Many data association algorithms have been proposed for use with SLAM. Generally speaking, batch data association algorithms (e.g. [1], [23]) are more robust than the standard maximum likelihood approach [8], but have a higher computational cost.

In D-SLAM, data association is required at two instances. One is for the local SLAM (Section 3.2.2); the other is for the mapping process in D-SLAM. For the local SLAM, the standard maximum likelihood approach is adequate. During the D-SLAM mapping process, the correlations between the robot and features are not maintained. Therefore, maximum likelihood hypotheses of possible associations, assuming that the correlations between the robot and features are zero, are first generated. These hypotheses are then evaluated using a chi-square test that compares the relative distances and angles between features computed using the measurement vector and the map state vector. While this does not require the correlation between the robot location and the features, it still needs the location estimates and the associated covariance matrix of the set of features that are potentially being observed. This feature set can be extracted from the current map using the current robot location estimate and the sensor range.

4.2 Recovery of feature locations and associated covariances

Recovery of the feature location estimates and the covariances associated with the features in the vicinity of the robot is needed for data association, map update and robot localization. When the number of features is small, these can be simply obtained by (7) using the inverse of the information matrix. However, when the number of features is large, the computational cost of the inversion will be unacceptable. Therefore, it is crucial to find an efficient method for state and covariance recovery.

For mapping, the locations of features that are being observed are required to compute ∇H_{map} and $\nabla H_{map}\hat{X}(k)$. The locations of previously seen features that are being observed and the associated covariance matrix are required for the localization step. For data association, location estimates and the associated covariance matrix of all the features that are within the sensor range are required.

Feature locations can be recovered by solving the sparse linear equation (7). Covariance recovery can also be done in a similar manner. Note that

$$I(k)P_i = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T \qquad (14)$$

where P_i is the ith column of the covariance matrix. Solving several linear equations of this form, columns of the covariance matrix which correspond to the features that are potentially being observed can be recovered. An efficient algorithm for this process is presented in Section 5.4.

4.3 Construction of admissible measurements

The D-SLAM algorithm described in the previous sections requires that the sensor observes multiple features, including at least two previously seen features at a given instant. This condition, while common with a high rate sensor such as a laser, may not be true when the feature density is low. In such situations, it is possible to combine a sequence of observations to construct an admissible measurement.

Figure 3 shows an example where the robot observes two previously seen features f_1, f_2 and two new

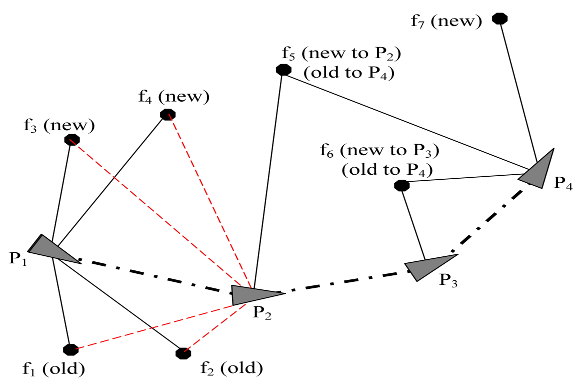


Figure 3: Construction of admissible measurements

features f_3, f_4 at point P_1, observes one new feature f_5 at point P_2, and one new feature f_6 at point P_3. Later on at point P_4, it observes features f_5, f_6, f_7. Thus the measurements at P_2 and P_3 are not admissible. It is, however, possible to combine the measurements made from different points to generate new admissible measurements as follows. Once it is detected that the observation at point P_2 is not admissible, the update to the map using the observation information from P_1 is removed. Then a virtual observation from P_2 to f_1, f_2, f_3, f_4 is constructed using the observation from P_1 to f_1, f_2, f_3, f_4 and an estimate of the relative motion of the robot from P_1 to P_2 (Figure 3). The uncertainty associated with this composite observation is computed using the relevant observation equations and the process and observation uncertainties. The mapping process can now proceed as if features f_1, f_2, f_3, f_4, f_5 are observed from P_2 and no observation is made at P_1. This process is repeated wherever an inadmissible observation is encountered, for example at P_3. This strategy allows D-SLAM to function where features are sparse or one cluster of features are separated from another cluster of features by a region of "featureless" terrain.

5 Computational complexity

Let N be the number of features in the map.

5.1 Storage

In two dimensional D-SLAM, storage is required for the information vector with dimension $2N$, the recovered state vector with dimension $2N$, the sparse information matrix with non-zero elements $O(N)$, and the columns of the covariance matrix corresponding to the features in the vicinity of the robot $O(N)$. The overall storage requirement is therefore $O(N)$.

5.2 Localization

Localization step in D-SLAM requires updating a state vector containing at most a constant number of elements, thus the computational cost is $O(1)$.

5.3 Mapping

Mapping in D-SLAM is formulated in the information form where the update step is an $O(1)$ time process. The prediction step, the computationally demanding stage of an information filter, does not exist.

5.4 State vector and covariance matrix recovery

The major computational cost of D-SLAM is, therefore, due to the need for recovering the feature location estimates and certain columns of the associated covariance matrix by solving a set of linear equations as discussed in Section 4.2.

5.4.1 Preconditioned Conjugated Gradient (PCG) for solving linear equations

When solving a linear equation $A_{n\times n}x_{n\times 1} = b_{n\times 1}$ using the Conjugated Gradient (CG) method, the computational cost of each iteration is dominated by a matrix-vector product, $A_{n\times n}d_{n\times 1}$, where d is an arbitrary vector updated in each iteration. Normally the product requires $O(n_z)$ operations, where n_z is the number of non-zero entries in matrix $A_{n\times n}$ (see [27]). In D-SLAM the matrix A is the sparse information matrix with each column (row) containing at most a constant number of non-zero elements (see Section 3.1.4), which means matrix A contains $O(N)$

non-zero elements. Thus, the matrix-vector multiplication that is necessary at each iteration of CG requires $O(N)$ operations, and the computational cost of each iteration in CG is $O(N)$.

In general, CG requires N iterations to converge, resulting in a total computational cost of $O(N^2)$. However, the number of iterations required for convergence can be substantially reduced using the preconditioned CG (PCG) method. With a good preconditioner, only a few (constant number) iterations are sufficient for PCG to converge. A good preconditioner itself requires significant amount of computation in general. However, the system matrix in the D-SLAM algorithm is the sparse information matrix and two consecutive information matrices are very similar due to the gradual evolution of the map. This special structure of the D-SLAM information matrix leads to an efficient method for the recursive computation of a good preconditioner based on approximate Cholesky Factorization.

5.4.2 Iterative procedure for preconditioning

Let $\nabla H_{map}^T R_{map}^{-1} \nabla H_{map}$ in equation (11) be expressed in the form

$$\nabla H_{map}^T R_{map}^{-1} \nabla H_{map} = \begin{bmatrix} 0 & 0 \\ 0 & H_R \end{bmatrix}. \tag{15}$$

The dimension of the square matrix H_R depends on which features are observed at time $k+1$. If all the observed features are present near the bottom of the current state vector, the dimension of H_R is of $O(1)$. On the other hand, if the observed feature set also contains a feature near the top of the state vector, the dimension of H_R is of $O(N)$.

The preconditioning process proposed is a function of the dimension of H_R.

Case (i). When the dimension of H_R is less than a threshold n_0 ($n_0 = 100$ is used in the simulation presented in Section 6.2 where the dimension of the final map state vector is $2N = 1192$), the approximate Cholesky Factorization of $I(k)$ is used to construct an approximate Cholesky Factorization of $I(k+1)$ as follows. Suppose the approximate Cholesky Factorization of $I(k)$ is $\tilde{L}_k$.

Let $\tilde{L}_k$ and $I(k)$ be partitioned based on (15) as

$$\tilde{L}_k = \begin{bmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{bmatrix}, \tag{16}$$

in which $\tilde{L}_{11}$ and $\tilde{L}_{22}$ are lower triangular matrices, and

$$I(k) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22} \end{bmatrix}. \tag{17}$$

Let $I(k+1)$ be partitioned as

$$I(k+1) = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}^{k+1} \end{bmatrix} = \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22} + H_R \end{bmatrix}. \tag{18}$$

As shown in the appendix, an approximate Cholesky Factorization of $I(k+1)$ can be obtained by

$$\tilde{L}_{k+1} = \begin{bmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22}^{k+1} \end{bmatrix} \tag{19}$$

where $\tilde{L}_{22}^{k+1}$ is an approximate Cholesky Factorization of the submatrix $H_R + \tilde{L}_{22}\tilde{L}_{22}^T$, which can be computed using incomplete Cholesky Factorization.

Case (ii). When the dimension of H_R is larger or equal to n_0, the state vector is reordered based on the distance from each feature to the current robot location. The feature that is furthest from the robot is placed at the top of the reordered state vector. The dimension of H_R that corresponds to the new state vector is now $O(1)$ and is a function of the sensor range. Once the information vector and the information matrix are reordered accordingly, an approximate Cholesky Factorization of the whole information matrix need to be computed to produce the appropriate preconditioner. This is because the preconditioner of the last step can not be used due to reordering. The process of computing the approximate Cholesky Factorization can also be implemented as incomplete Cholesky Factorization.

The computational cost in Case (i) is $O(1)$ since the size of the submatrix $H_R + \tilde{L}_{22}\tilde{L}_{22}^T$ is less than a constant $n_0 \times n_0$. Case (ii) involves reordering the state vector, the information vector and the associated information matrix, together with an approximate Cholesky Factorization of the newly ordered sparse information matrix. Therefore Case (ii) will be more computationally expensive as the iterative process described previously can not be used to compute the preconditioner. However, Case (ii) only occurs occasionally [1] as the state vector is reordered according to distance and Case (i) will apply until the robot travels a large distance and observes a feature near the top of the state vector.

Although the approximate Cholesky Factorization may introduce fill-in, simulation results show that the reordering of the state vector according to distance as described in Case (ii) significantly reduces the number of fill-in. This reordering has a similar effect as shown in [7]. Simulation results presented in Section 6.2 demonstrate that the computational cost of PCG is $O(N)$ (Figure 6(d)).

5.4.3 Recovery of the covariance matrix

Columns of the covariance matrix, which correspond to the features in the vicinity of the current robot location, are required. A column of the covariance matrix can be computed with a cost similar to that of computing the state vector as the same preconditioner can be used in this process. Therefore the computational cost of recovering a constant number of columns of the covariance matrix is still $O(N)$.

5.4.4 Initial guesses

Linear equation solvers require initial guess of the solution vector. Once the preconditioner is well chosen, initial guesses do not significantly influence the number of iterations necessary. In the extreme case when exact Cholesky Factorization is used as the preconditioner, PCG converges in one step from any initial guess. As the features are stationary, the previous estimates provide good initial guesses for the feature locations. During the experiments and computer simulations, it was seen that zeros appear to be adequate as initial guesses for the elements of the covariance matrix.

[1]The number of times Case (ii) can occur depends on the parameter n_0, the sensor range, the density of features and the robot trajectory. If there is no loop closure, Case (ii) will never happen. In the simulation presented in Section 6.2 which contains many loop closures, Case (ii) occurs in 73 out of total 7876 loops.

6 Evaluation of D-SLAM

6.1 Experimental evaluation with an indoor robot

The Pioneer 2 DX robot was used for the experimental evaluation. This robot is equipped with a laser range finder with a field of view of 180 degrees and an angular resolution of 0.5 degree. Test site was in our laboratory where twelve laser reflector strips were placed in an $8 \times 8m^2$ area. The Player software [16] was used to collect the control and sensor data from the robot. A Matlab (Mathwork inc., U.S.A.) implementation of D-SLAM was used to process the data and compute the robot and feature locations.

Figure 4(a) shows the map obtained from D-SLAM. Figure 4(b) shows the error in the robot location estimate from D-SLAM with respect to that from the traditional SLAM algorithm, which is used as baseline in the comparison. Figures 4(c) and 4(d) show the 2σ error bounds obtained from D-SLAM and traditional SLAM for the estimates of robot location and feature 9 respectively.

Figure 4(b) shows that the localization error in D-SLAM falls within the 2σ error bounds, and thus the estimation is consistent. The map (Figure 4(a)) is almost as good as that of the traditional SLAM in this small area, as can be seen more clearly in Figure 4(d). In this figure, the 2σ error bounds from D-SLAM are very close to that from traditional SLAM.

It can be seen from Figure 4(c) that the localization result using CI is conservative compared with that from traditional SLAM, as expected. The difference between the 2σ error bounds from D-SLAM and those from traditional SLAM in robot location and orientation estimation are less than 0.05 m and 0.02 rad respectively. Compared with the size of the environment, this is small and acceptable.

6.2 Evaluation using a large-scale simulation

A more complex simulation experiment using a large number of features was conducted to further evaluate D-SLAM and demonstrate its properties. The environment used is a 100 meter square with 1225 features arranged in uniformly spaced rows and columns. The robot starts from the bottom left corner of the square and follows a random trajectory, revisiting many features and closing many loops as seen in Figures 5(a) and 5(b). 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.

Figures 5(a) and 5(b) show the robot trajectory and feature location estimates from traditional EKF SLAM and D-SLAM respectively. The difference between the feature location estimates from the two algorithms are not distinguishable in the figures. Figures 5(c) and 5(d) show robot location estimates from traditional EKF SLAM and D-SLAM respectively. The D-SLAM result is consistent in the sense that most of the estimation errors fall within the 2σ error bounds. It can also be noticed that the D-SLAM result is conservative.

Figure 5(e) shows the estimation error and the associated 95% confidence levels for feature 7. It is clear that the estimates are consistent. Figure 5(f) demonstrates the standard deviation of the location estimates for a set of randomly selected features. It shows that the uncertainty of the feature location estimates decreases monotonically. At around 600 seconds, all feature location estimates are improved as a loop is closed.

Figure 6(a) shows the links among the features in the final information matrix. Figure 6(b) shows all the non-zero elements of the information matrix in black. This information matrix is obtained without using the iterative preconditioning technique in Section 5.4.2. It is clear that this matrix is sparse as there are 44692 non-zero elements and 1376172 exactly zero elements. The block diagonal areas are due to features in close vicinity observed together and the off-diagonal terms are due to loop closures where a previously seen feature is reobserved some time later. The information matrix in Figure 6(c) is obtained using the iterative preconditioning technique in Section 5.4.2. The information matrix is banded because of the reordering of the state vector. This matrix demonstrates the fact that only the nearby features are linked in the information matrix.

In the simulation, the approximate Cholesky Factorization of the submatrix $H_R + \tilde{L}_{22}\tilde{L}_{22}^T$ in Case (i) and that of the reordered information matrix in Case (ii) in Section 5.4.2 is implemented as incomplete Cholesky Factorization using MATLAB buildin function “cholinc” with the drop tolerance 10^{-7}. Figure 6(d) shows the time required to recover the state vector and one column of the covariance matrix using PCG together with the time required to compute the preconditioner using the iterative process described in Section 5.4.2 as a function of the number of features. Average of the computational time is used in cases where there are many steps for which the number of features is identical. The ratio between the computational cost and the number of features is seen to be a constant when the number of features is large. This indicates that the computational cost of PCG (recovering the state and/or one column of the covariance matrix) and preconditioning are both $O(N)$ for large maps. Figure 6(e) shows the average number of PCG iterations as a function of the number of features. In general it takes at most 2 iterations to converge.

Figure 6(f) compares the average time for Cholesky Factorization using direct method used in [7] for incremental $\sqrt{SAM}$ and the average time for Cholesky Factorization using the iterative method described in Section 5.4.2. Both times are shown as a function of the number of features. It is clear that the computational saving achieved by the iterative method is significant.

6.3 Outdoor experimental evaluation using Victoria Park data set

This large-scale outdoor data set, available from Australian Centre of Field Robotics (ACFR) [22], is collected by a standard utility vehicle which is fitted with dead reckoning sensors and a laser range finder. Information from the laser is processed to extract location of the trees in the park. Ground truth for this dataset is not available. Therefore it is only possible to comment on the outputs of the two algorithms. It is important to note that during this practical example, D-SLAM and traditional SLAM behave slightly differently as feature validation and data association strategies are specific to the algorithm being implemented. Therefore some features present in the traditional SLAM map are not present in the D-SLAM map and vice versa.

The estimates of the feature positions and the vehicle trajectories obtained by the D-SLAM algorithm and the traditional EKF SLAM algorithm are shown in Figure 7(a). The maximal difference between the two trajectories is around $3m$.

Figures 7(b) shows the standard deviations of the vehicle position estimates obtained from D-SLAM and traditional SLAM. It is seen, as expected, that the estimates of the errors from D-SLAM are always higher than those from traditional SLAM but the differences are clearly not very significant.

Figure 7(c) compares the uncertainties of the location estimates for two features. These features correspond to the two extremes in terms of difference in final feature location estimation uncertainty between the two algorithms: one corresponds to the smallest difference, and the other corresponds to the largest. Again the estimates of the errors from D-SLAM are always above those from traditional SLAM, confirming that D-SLAM is conservative. However, even in the worst case scenario, the difference in estimation uncertainty is less than $1m$.

Figure 7(d) shows the sparse information matrix of D-SLAM result. In this scenario, the sensor range is significant with respect to the size of the environment. The information matrix is, therefore, not very sparse. In particular, the high density of non-zero elements in the top-left part of the matrix is caused by many loop closing events near the starting point of the vehicle.

6.4 Evaluation of information loss

The fact that the information in z_{rob} is not used in mapping and that the full state vector that includes the robot and feature locations are not used in the estimation process clearly results in some information loss. This can also be seen from the results shown in the 2D simulation and experimental results, where the covariance ellipses in the case of D-SLAM are larger than those obtained using the traditional SLAM algorithm.

An analysis of the extent of information loss of the D-SLAM mapping process based on a linear one-dimensional form of the SLAM problem is presented below. The 1D simulation is used so that the effects due to linearization errors are avoided.

The scenario consists of a set of uniformly distributed features arranged on a straight line. The robot, moving forward and backward along this line, can measure the distances to features within its sensor range ($9m$). Four parameters are considered: the process noise, the sensor noise, the feature density, and the number of observations made before the robot starts moving away from its initial position. The trace of the submatrix of the covariance matrix corresponding to all features from traditional SLAM, P_{mm}, and the trace of the covariance matrix from D-SLAM, P, are used as indicators of the filter performance. Their ratio is used to evaluate the information loss of D-SLAM. Table 1 summarizes the results obtained.

The one dimensional analysis shows the following with respect to information loss.

(1) When all other parameters are fixed, the larger the process noise, the smaller the amount of information lost (compare rows $2-4$ with row 1 in Table 1). In the limit when the process noise goes to infinity (row 4 in Table 1), D-SLAM and traditional EKF SLAM produce identical maps. This is expected as D-SLAM mapping proceeds as if there is no information on the robot location available. However, this interesting result can not be verified for 2D case as 2D traditional EKF SLAM will either diverge or provide inconsistent results when the process noise is too large. D-SLAM is, however, feasible provided that data association is available.

(2) When all other parameters are fixed, the larger the observation noise, the larger the extent of information loss (compare rows $5-6$ with row 1 in Table 1). This is because, for mapping, D-SLAM only exploits the information contained in the observations. When the ratio between observation noise and process noise is high, the amount of information present in the observations is comparatively smaller.

(3) When all other parameters are fixed, the larger the number of observations made before robot moves away from its initial position, the smaller the extent of information loss in D-SLAM (compare rows $7-8$

Test No.	Process noise (m/s)	Sensor noise (m)	Interval of features (m)	Loops before robot moves	EKF SLAM trace of P_{mm}	D-SLAM trace of P	Trace ratio $\frac{D-SLAM}{EKF\ SLAM}$
1	**0.02**	**0.07**	**3**	**15**	**0.00135**	**0.00176**	**1.304**
2	***0.1***	0.07	3	15	0.00167	0.00176	1.054
3	***0.2***	0.07	3	15	0.00172	0.00176	1.023
4	***2000***	0.07	3	15	0.00176	0.00176	1.000
5	0.02	***0.7***	3	15	0.06310	0.17600	2.789
6	0.02	***7***	3	15	4.96373	17.60015	3.546
7	0.02	0.07	3	***10***	0.00146	0.00206	1.411
8	0.02	0.07	3	***5***	0.00174	0.00316	1.816
9	0.02	0.07	***2***	15	0.00137	0.00162	1.182
10	0.02	0.07	***1***	15	0.00362	0.00398	1.099

Table 1: Analysis of information loss in D-SLAM using 1D simulations

with row 1 in Table 1). This is due to the fact that when the robot is stationary at its initial position there is no information loss in D-SLAM.

(4) When all other parameters are fixed, the larger the feature density in the environment, the smaller the extent of information loss (compare rows $9-10$ with row 1 in Table 1). This is because the ratio of the dimension of z_{map} $(2m-3)$ and z_{old} $(2m)$ is closer to 1 when robot can observe more features (m) in each scan. Therefore, comparatively, there is more information in z_{map} than in z_{rob} and the robot process model.

Despite the fact that there is some information loss, the results shown earlier in this section demonstrate that D-SLAM provides significant computational advantages and good quality estimates of the map and robot location in representative practical scenarios.

7 Conclusions and Future Work

In this paper, a new decoupled SLAM algorithm: D-SLAM, is described. While the localization and mapping are performed simultaneously, they are separate processes. This new algorithm is based on a method to recast the observation vector such that the information about the map is extracted independent of the robot location.

Although the robot location is not incorporated in the state vector used in mapping, correlations among the features are still preserved. Thus the location estimates of all the features are improved using information from one local observation.

The significant advantages gained are that there is no prediction step for the mapping, the information matrix associated with mapping is exactly sparse and only the features that are in the close vicinity are linked through the information matrix. This results in an $O(N)$ SLAM algorithm where N is the number of features.

In D-SLAM, however, the knowledge about the robot location is not exploited in the mapping process. This results in some information loss. An analysis based on a linear one-dimensional simulation indicated that the ratio between the sensor noise and the process noise is one of the key factors influencing the extent of the information loss. The smaller this ratio, the smaller the amount of information loss. However, the extent of information loss for 2D scenarios was found to be difficult to appropriately quantify.

D-SLAM has some similarities with the very recent Exactly Sparse Extended Information Filter (ESEIF) by Walter et al. [31]. Both these methods achieves exactly sparse information matrix without any approximation and without including the robot trajectory in the state vector. The main differences between ESEIF and D-SLAM are: in D-SLAM, the robot is not in the map state vector and the level of sparseness of the information matrix naturally depends on the sensor range and feature density; while in ESEIF, the robot is still a part of the state vector, and the information matrix needs to be controlled to achieve sparseness by deliberately "kidnapping robot" and "relocating robot" from time to time. Both algorithms have some information loss as compared with traditional EKF SLAM. The main factor influencing the extent of information loss in ESEIF is how often the robot is kidnapped and relocated. Which of these two algorithms results in more information loss depends on the specific parameters used, the motion of the robot and the environment.

The computational cost $O(N)$ in each step may not be acceptable for very large-scale SLAM problems. Applying D-SLAM in conjunction with the submaps (e.g. [29], [32]) has the potential to further reduce the computational effort. By using the submaps [17], it is also possible to deal with the bearing-only and range-only SLAM problems. The research along this direction is underway.

Some recent results have shown that large errors in the robot orientation introduce inconsistency in traditional SLAM [3], [15]. D-SLAM does not have the robot location in the state vector used for mapping thus may be more robust than traditional SLAM. The study on the potential robustness of D-SLAM against the heading error of robot will also be very interesting.

Acknowledgment

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.

References

[1] Bailey, T. 2002. Mobile robot localization and mapping in extensive outdoor environment. Ph.D. Thesis. Australian Centre of Field Robotics, University of Sydney.

[2] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. 2001. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software.* John Wiley & Sons. (Electronic Version)

[3] Castellanos, J. A., Neira. J., and Tardos, J. D. 2004. Limits to the consistency of EKF-based SLAM. *Proc. 2004 IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal.

[4] Chen, L., Arambel, P. O., and Mehra, R. K. 2002. Estimation under unknown correlation: covariance intersection revisited. *IEEE Transactions on Automatic Control* 47(11):1879-1882.

[5] Csorba, M., Uhlmann, J. K., and Durrant-Whyte, H. 1997. A suboptimal algorithm for automatic map building. In *Proc. American Control Conference*, pp. 537-541.

[6] Deans, M. C., and Hebert, M. 2000. Invariant filtering for simultaneous localization and map building. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 1042-1047.

[7] Dellaert, F. 2005. Square root SAM. In *Proc. Robotics: Science and Systems.* See webpage: http://www.roboticsproceedings.org/rss01/ index.html.

[8] Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., and Csobra, M. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation* 17(3):229-241.

[9] Eustice, R. M., Singh, H., and Leonard, J. 2005. Exactly sparse delayed-state filters. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 2428-2435.

[10] Eustice, R. M., Walter, M., and Leonard, J. 2005. Sparse extended information filters: insights into sparsification. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 641-648.

[11] Eustice, R., Singh, H., Leonard, J., Walter, M., and Ballard, R. 2005. Visually navigating the RMS Titanic with SLAM information filters. In *Proc. Robotics: Science and Systems.* See webpage: http://www.roboticsproceedings.org /rss01/p08.html.

[12] Folkesson, J., and Christensen, H. I. 2004. Graphical SLAM - a self-correcting map. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 383-390.

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

[14] Frese, U., Larsson, P., and Duckett, T. 2005. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics* 21(2):196-207.

[15] Frese, U. 2006. A discussion of simultaneous localization and mapping. *Autonomous Robots* 20 (1):25-42.

[16] Gerkey, B. P., Vaughan, R. T., and Howard, A. 2003. The player/stage project: Tools for multirobot and distributed sensor systems. In *Proc. International Conference on Advanced Robotics*, pp. 317-323.

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

[18] Julier, S. J., and Uhlmann, J. K. 2001. Simultaneous localization and map building using split covariance intersection. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1257-1262.

[19] Martinelli, A., Tomatics, N., and Siegwart, R. 2004. Open challenges in SLAM: An optimal solution based on shift and rotation invariants. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 1327-1332.

[20] Maybeck, P. 1979. *Stochastic Models, Estimation, and Control. Vol.1.* Academic, New York.

[21] Moutarlier, P., and Chatlia, R. 1989. Stochastic multisensor data fusion for mobile robot localization and environment modeling. In *Proc. International Symposium on Robotics Research*, pp. 85-94.

[22] Nebot, E. M. UTE Experimental Data from Victoria Park. See webpage: http://www.acfr. usyd.edu.au/homepages/academic/enebot/ experimental_data_ute.htm.

[23] Neira, J., and Tardos, J. D. 2001. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions Robotics and Automation* 17(6):890-897.

[24] Newman, P. 2000. On the structure and solution of the simultaneous localization and map building problem. Ph.D. Thesis. Australian Centre of Field Robotics, University of Sydney.

[25] Pradalier, C., and Sekhavat, S. 2003. Simultaneous localization and mapping using the Geometric Projection Filter and correspondence graph matching. *Aadvanced Robotics* 17(7):675 - 690.

[26] Saad, Y. 1996. Iterative Methods for Sparse Linear Systems. PWS Publishing Company. (Electronic Version)

[27] Shewchuk, J. 1994. An Introduction to the Conjugate Gradient Method without the Agonizing Pain. Technical Report. CMU-CS-94-125, Carnegie Mellon Univeristy, Pittsburgh, PA, USA.

[28] Smith, R., Self, M., and Cheeseman, P. 1990. Estimating uncertain spatial relationships in robotics. In *Automomous Robot Vehicles*, Cox, I. J., and Wilfon, G. T., eds, New York: Springer Verleg. pp. 167-193.

[29] Tardos, J. D., Neira, J., Newman, P., and Leonard, J. 2002. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research* 21(4):311-330.

[30] Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. 2004. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research* 23(7-8):693-716.

[31] Walter, M., Eustice, R., and Leonard, J. 2005. A provably consistent method for imposing exact sparsity in feature-based SLAM information filters. In *Proc. International Symposium on Robotics Research.* See webpage: http://robot.cc/program.html.

[32] Williams, S. B. 2001. Efficient solutions to autonomous mapping and navigation problems. Ph.D. Thesis. Australian Centre of Field Robotics, University of Sydney.

A Iterative Method for Cholesky Factorization

In this appendix, it is shown that the Cholesky Factorization of the information matrix $I(k+1)$ can be constructed from that of $I(k)$ due to the similarity between the two consecutive information matrices as shown in (11).

A.1 Cholesky Factorization

Suppose the Cholesky Factorization of $I(k)$ is L_k (a lower triangular matrix). Then

$$I(k) = L_k L_k^T. \tag{20}$$

Let L_k and $I(k)$ be partitioned according to (15) as

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

Then from (20),

$$\begin{aligned} L_{11}L_{11}^T &= I_{11}, \\ L_{21}L_{11}^T &= I_{21}, \\ L_{21}L_{21}^T + L_{22}L_{22}^T &= I_{22}. \end{aligned} \tag{22}$$

According to (11), (15) and (21), $I(k+1)$ can be expressed by

$$\begin{aligned} I(k+1) &= \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}^{k+1} \end{bmatrix} \\ &= \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}+H_R \end{bmatrix}. \end{aligned} \tag{23}$$

Lemma A.1 *The Cholesky Factorization of $I(k+1)$ is*

$$L_{k+1} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22}^{k+1} \end{bmatrix} \tag{24}$$

where L_{22}^{k+1} is the Cholesky Factorization of the submatrix $H_R + L_{22}L_{22}^T = I_{22}^{k+1} - L_{21}L_{21}^T$. That is,

$$\begin{aligned} L_{22}^{k+1}(L_{22}^{k+1})^T &= H_R + L_{22}L_{22}^T \\ &= I_{22}^{k+1} - L_{21}L_{21}^T. \end{aligned} \tag{25}$$

Proof: By (22), (24) and (25),

$$\begin{aligned} & L_{k+1}L_{k+1}^T \\ =& \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22}^{k+1} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & (L_{22}^{k+1})^T \end{bmatrix} \\ =& \begin{bmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}^{k+1}(L_{22}^{k+1})^T \end{bmatrix} \\ =& \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & L_{21}L_{21}^T + H_R + L_{22}L_{22}^T \end{bmatrix} \\ =& \begin{bmatrix} I_{11} & I_{21}^T \\ I_{21} & I_{22}+H_R \end{bmatrix}. \end{aligned} \tag{26}$$

Thus by (23),

$$I(k+1) = L_{k+1}L_{k+1}^T. \qquad (27)$$

Since both L_{11} and L_{22}^{k+1} are lower triangular, L_{k+1} is also lower triangular. Thus L_{k+1} is the Cholesky Factorization of $I(k+1)$.

Remark: When the dimension of H_R in (15) is $O(1)$, the computational cost of computing L_{k+1} using (24) is $O(1)$, which is much more efficient than directly computing the Cholesky Factorization of $I(k+1)$.

A.2 Approximate Cholesky Factorization

The formula (24) in the above section can also be used to iteratively compute an approximate Cholesky Factorization of $I(k+1)$.

Suppose $\tilde{L}_k$ is an approximation of Cholesky Factorization of $I(k)$, then

$$I(k) \approx \tilde{L}_k\tilde{L}_k^T. \qquad (28)$$

Let $\tilde{L}_k$ be partitioned according to (15) as

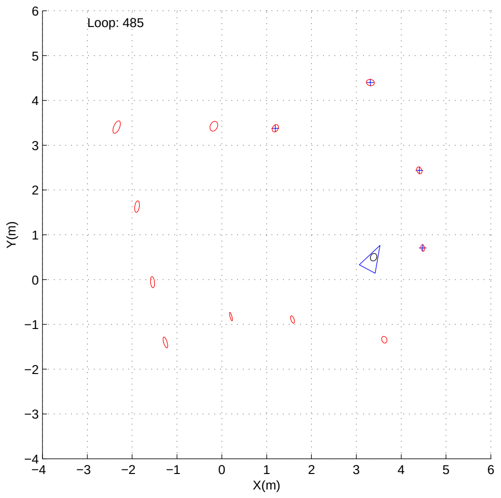
$$\tilde{L}_k = \left[\begin{array}{cc} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{array}\right]. \qquad (29)$$

Let $\tilde{L}_{22}^{k+1}$ be an approximate Cholesky Factorization of the submatrix $H_R + \tilde{L}_{22}\tilde{L}_{22}^T$ and construct $\tilde{L}_{k+1}$ by

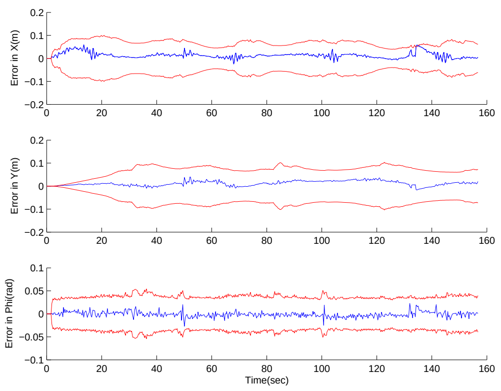
$$\tilde{L}_{k+1} = \left[\begin{array}{cc} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22}^{k+1} \end{array}\right]. \qquad (30)$$

Then the following approximate equation can be proved in a way similar to the proof of Lemma A.1,

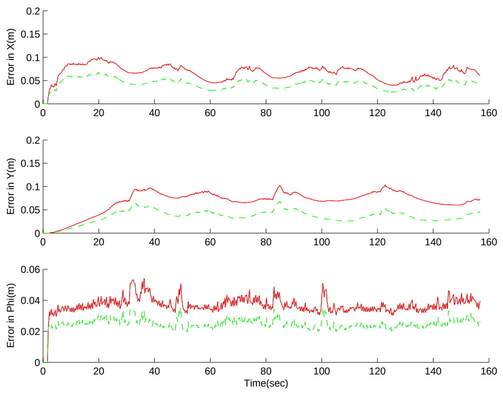
$$I(k+1) \approx \tilde{L}_{k+1}\tilde{L}_{k+1}^T. \qquad (31)$$



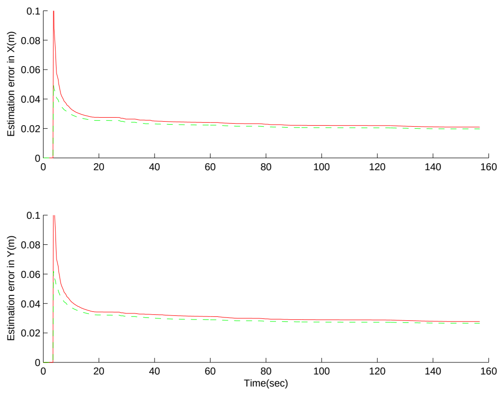
(a) Map obtained by D-SLAM (ellipses indicate 2σ error bounds for feature and robot location estimates)



(b) Error of the robot location estimate and associated 2σ error bounds

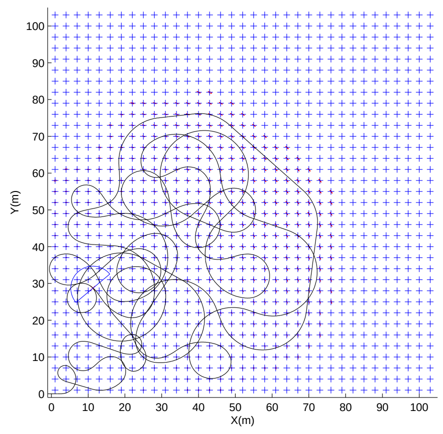


(c) 2σ error bounds of robot location estimate (solid line is from D-SLAM; dashed line is from traditional SLAM)

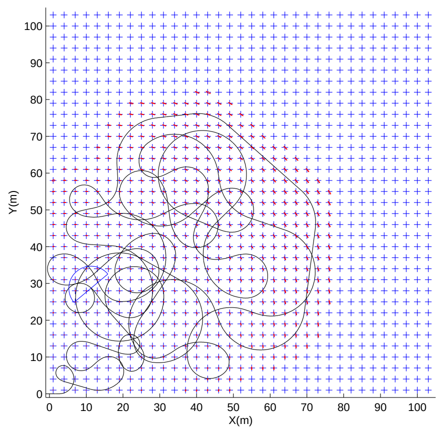


(d) 2σ error bounds for the location estimate of feature 9 (solid line is from D-SLAM; dashed line is from traditional SLAM)

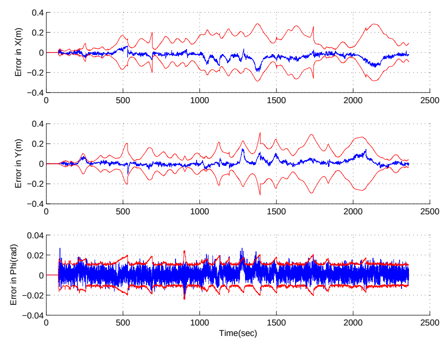
Figure 4: Implementation of D-SLAM in an indoor environment



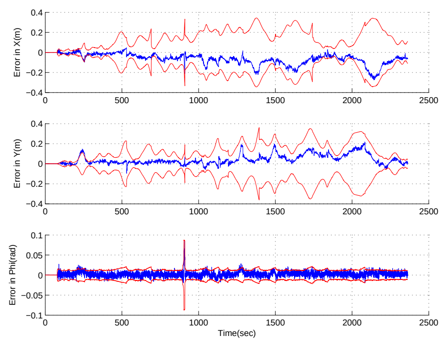
(a) Robot trajectory and the map obtained by traditional EKF SLAM



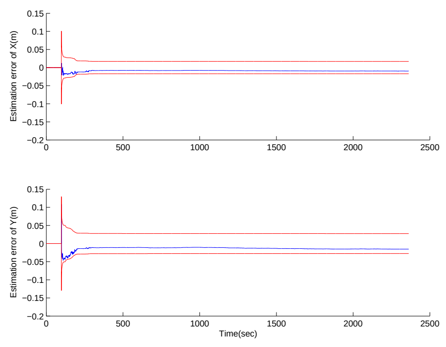
(b) Robot trajectory and the map obtained by D-SLAM



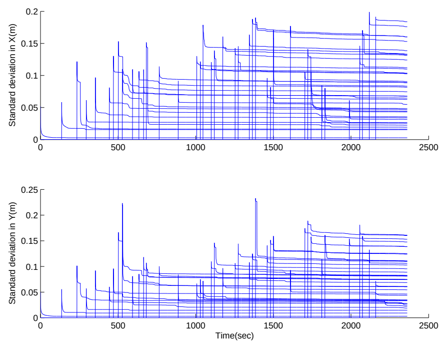
(c) Robot location estimation errors and 2σ error bounds from traditional EKF SLAM



(d) Robot location estimation errors and 2σ error bounds from D-SLAM

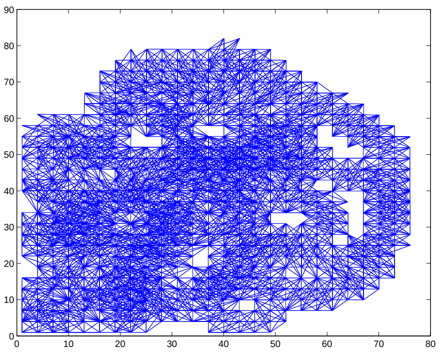


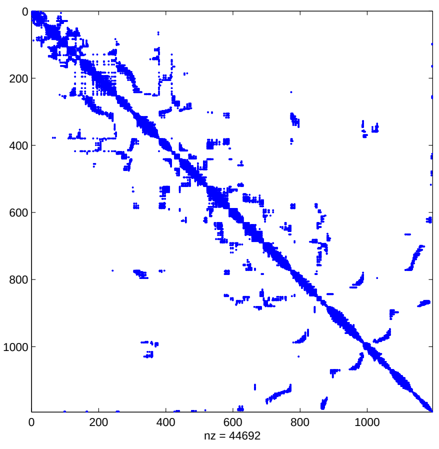
(e) Estimation error of feature 7 from D-SLAM



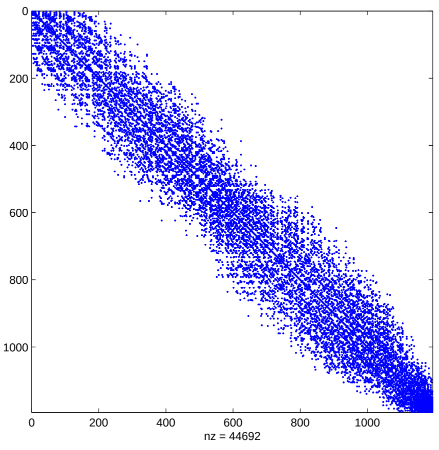
(f) Standard deviation of location estimates of several randomly selected features from D-SLAM

Figure 5: Simulation results with large number of features

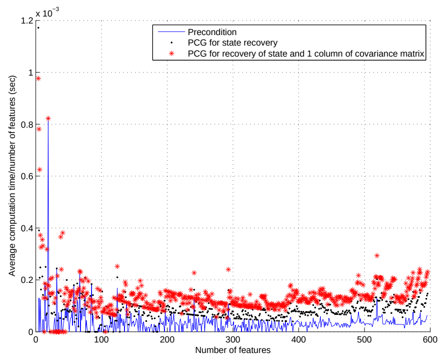

(a) Links in the information matrix



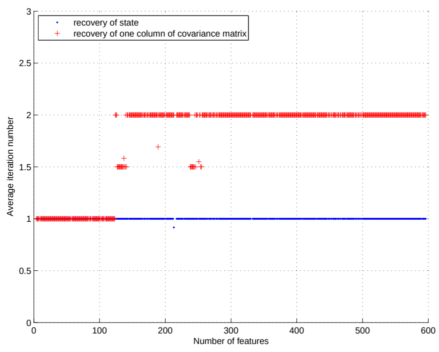
(b) Sparse information matrix obtained by D-SLAM without any reordering of the state vector



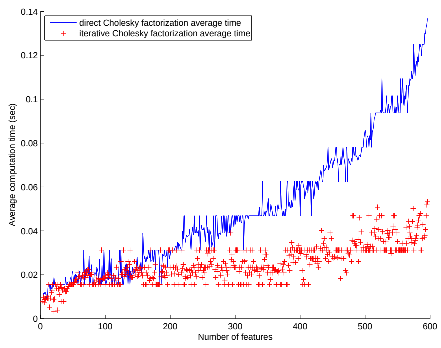
(c) Banded information matrix obtained by D-SLAM with reordering of the state vector



(d) Average time for precondition and PCG divided by the number of features

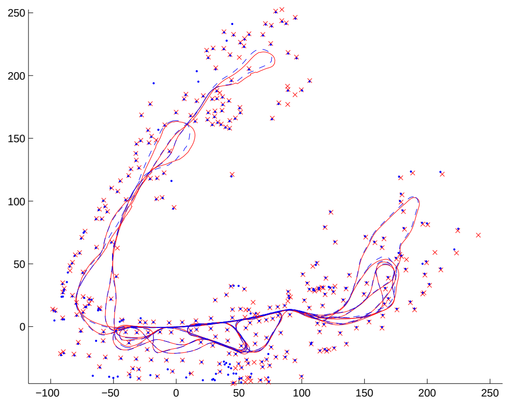


(e) Average iteration number for PCG convergence (previous state vector is used as the initial guess for recovering current state vector; zeros are used as the initial guesses for recovering columns of covariance matrix)

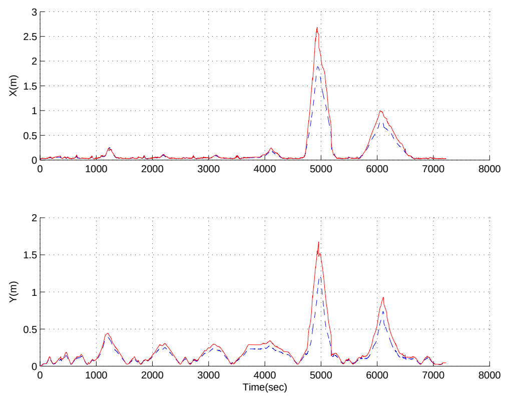


(f) Comparison of the average time required for the Cholesky Factorization using the iterative method and the direct Cholesky Factorization

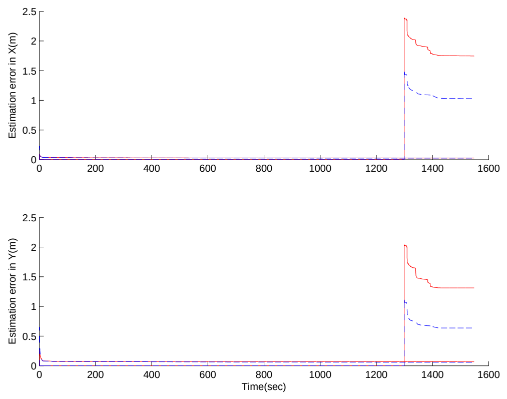
Figure 6: D-SLAM simulation results – sparse information matrix and computation time



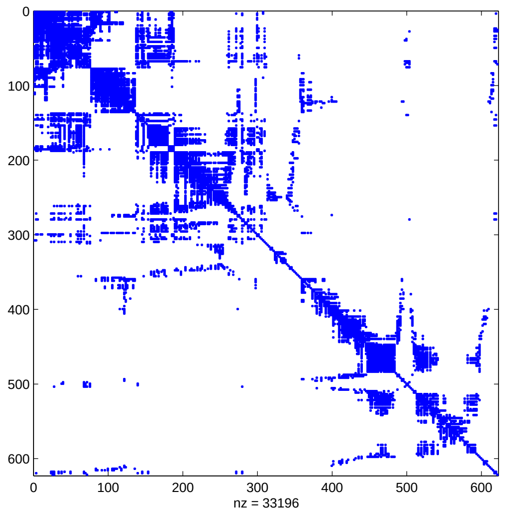
(a) Map and vehicle trajectory (solid line and cross are trajectory and feature location from D-SLAM; dashed line and dot are trajectory and feature location from traditional EKF SLAM)



(b) 1σ error bounds for robot position estimate (solid line is from D-SLAM; dashed line is from traditional EKF SLAM)



(c) 2σ error bounds for the feature location estimate of the two extreme cases in terms of difference in resulting estimation uncertainty (solid line is from D-SLAM; dashed line is from traditional EKF SLAM)



(d) Exactly sparse information matrix (33196 nonzero elements and 353688 exactly zero elements)

Figure 7: Outdoor, large-scale implementation of D-SLAM using Victoria Park data set