Linear SLAM: Linearising the SLAM Problems using Submap Joining ⋆

Liang Zhao, Shoudong Huang, Gamini Dissanayake

Centre for Autonomous Systems, Faculty of Engineering and Information Technology
University of Technology Sydney, Australia

Abstract

The main contribution of this paper is a new submap joining based approach for solving large-scale Simultaneous Localization and Mapping (SLAM) problems. Each local submap is independently built using the local information through solving a small-scale SLAM; the joining of submaps mainly involves solving linear least squares and performing nonlinear coordinate transformations. Through approximating the local submap information as the state estimate and its corresponding information matrix, judiciously selecting the submap coordinate frames, and approximating the joining of a large number of submaps by joining only two maps at a time, either sequentially or in a more efficient Divide and Conquer manner, the nonlinear optimization process involved in most of the existing submap joining approaches is avoided. Thus the proposed submap joining algorithm does not require initial guess or iterations since linear least squares problems have closed-form solutions. The proposed Linear SLAM technique is applicable to feature-based SLAM, pose graph SLAM and D-SLAM, in both two and three dimensions, and does not require any assumption on the character of the covariance matrices. Simulations and experiments are performed to evaluate the proposed Linear SLAM algorithm. Results using publicly available datasets in 2D and 3D show that Linear SLAM produces results that are very close to the best solutions that can be obtained using full nonlinear optimization algorithm started from an accurate initial guess. The C/C++ and MATLAB source codes of Linear SLAM are available on OpenSLAM.

Key words: SLAM; linear least squares; submap joining; feature-based SLAM; pose graph SLAM; D-SLAM.

1 Introduction

Simultaneous Localization and Mapping (SLAM) is the problem of using a mobile robot to build a map of an unknown environment and at the same time locating the robot within the map [1, 2]. Recently, nonlinear optimization techniques have become popular for solving SLAM problems. Following the seminal work by Lu and Milios [3], many efficient SLAM solutions have been developed by exploiting the sparseness of the information matrix and applying different approaches for solving the sparse linear equations [4–8]. However, since SLAM is formulated as a high dimensional nonlinear optimization problem, finding the global minimum is nontrivial. Although many SLAM algorithms appear to provide good results for most of the practical datasets (e.g. [9–11]),

there is no guarantee that the algorithm can converge to the global optimum and having an accurate initial value is very critical, especially for large-scale SLAM problems. Moreover, for very large-scale SLAM problems (e.g. life long SLAM), it is in general intractable to keep all the original data and perform full nonlinear optimization to obtain the SLAM solution. Some sort of information fusion is necessary to reduce the computational complexity.

Local submap joining has shown to be an efficient way to build large-scale maps [12–18]. The idea of many map joining algorithms such as Sparse Local Submap Joining Filter (SLSJF) [15] is to treat the estimated state of each local map as an integrated observation (the uncertainty is expressed by the local map covariance matrix) in the map joining step. By summarising the original data within the local map in this way, the SLAM problem can be solved more efficiently. It should be noted that most of the existing map joining problems are formulated as nonlinear estimation or nonlinear optimization problems which are solved by Extended Kalman Filter (EKF) [12–14], Extended Information Filter (EIF) [15] or nonlinear least squares [16–19] techniques.

This paper provides a new map joining framework which only requires solving linear least squares problems and performing nonlinear coordinate transformations. The method can be applied to join pose-feature maps (maps that contain both robot poses and features), pose-only maps (maps that only contain robot poses), and feature-only maps (maps that only contain features), for both 2D and 3D scenarios. There is no assumption on the structure of the covariance matrices of the local maps (no need to be spherical as in [20–22]). Since linear least squares problems have closed-form solutions, there is no need of an initial guess and no need of iterations to solve the reformulated map joining problems. It is demonstrated using publicly available datasets that the proposed Linear SLAM algorithm can provide accurate SLAM results. For practical applications, the proposed algorithm can be easily combined with other robust local map building algorithms in complex unknown environments such as [23, 24] to solve practical large-scale SLAM problems in a more robust manner.

The paper is an extended version of our preliminary work [26]. The major improvements of this paper over [26] are: (1) It has been shown that Linear SLAM is also applicable to the joining of feature-only maps; thus it provides a unified framework for solving different versions of SLAM problems (pose-feature, pose-only, and feature-only); (2) The consistency analysis of Linear SLAM is performed to demonstrate its estimation consistency; (3) The theoretical analysis on the computational complexity of Linear SLAM is also performed to demonstrate its efficiency; (4) More quantitive comparisons and evaluations of Linear SLAM are performed; (5) More insights and discussions are provided; and (6) More efficient C/C++ implementation of the algorithm is done and the source code is available on OpenSLAM.

The paper is organized as follows. Section 2 discusses the related work. Section 3 points out that different local maps can be built using the same SLAM data by selecting different coordinate frames. Section 4 explains the process of using linear least squares to solve the problem of joining two pose-feature maps. Section 5 explains how to use the linear method in the sequential and Divide and Conquer local submap joining. Section 6 and Section 7 describe how to apply the linear algorithm to the joining of two pose-only maps and two feature-only maps, respectively. In Section 8, simulation and experimental results using publicly available datasets are given to demonstrate the accuracy of Linear SLAM, and compare with some other submap joining based SLAM algorithms. Section 9 discusses the pros and cons of the proposed algorithm. Finally, Section 10 concludes the paper. To improve the readability of the paper, some technical details are provided in the appendices.

2   Related Work

Local submap joining has been a strategy applied by many researchers [12–19].

The earlier work on map joining are proposed by Tardos *et al.* (Map joining 1.0) [13] and Williams [14]. Both of them apply sequential map joining where a global map and a local map are joined at a time. In Map joining 1.0 [13], the idea used in joining two maps is to first transform one of the map using an estimate of relative pose from another map to have both submaps in a common reference, and then apply the coincidence constraint to the common elements. Williams [14] uses an EKF to fuse the local maps sequentially where each local map is treated as an integrated observation. Since the global map becomes larger and larger after sequentially fusing the local maps, the dense covariance matrix makes the EKF algorithm less efficient. To improve the efficiency of the map joining process, SLSJF is proposed in [15] where EIF is used and exactly sparse information matrix is achieved by keeping the robot starting poses and end poses of the local maps in the global state vector. CF-SLAM proposed in [35] further demonstrated that by using the Divide and Conquer strategy on top of EIF, the efficiency (and in most cases accuracy) could be improved further.

To overcome the potential estimate inconsistency of EKF/EIF in map joining, I-SLSJF is proposed in [19], which is an optimization based map joining treating each local map as an integrated observation, and using the result of SLSJF as the initial value for the nonlinear least squares optimization. More optimization based map joining are proposed in the past few years, for both feature-based SLAM [18, 39] and pose graph SLAM [17, 38, 40]. In [18], the local maps without any common nodes are first optimized independently, and then the base node of each local map is optimized by using the inter-measurements between different local maps, where the linearization of the local maps can be cached and reused. While in [39], after building the local maps, the condensed factors are generated based on the solution of the local maps, where the solution to the new graph is a global configuration of the origins and the shared variables. Then a good initial guess can be determined from the solutions of the origins and the shared variables, and the full nonlinear least squares is performed to get the exact results. In [38], a hierarchical pose graph optimization on manifolds is proposed, where only the coarse structure of the scene is corrected, resulting in an efficient SLAM algorithm. In [40], a closed-form online 3D pose-chain SLAM algorithm is proposed. Due to the spherical assumption on the covariance matrices, it can efficiently optimize pose-chains by exploiting their Lie group structure. The spherical covariance assumption significantly reduces the complexity of the SLAM problems, as shown in recent works [21, 22, 25]. In [17], a relative formulation of

the relationship between multiple pose graphs is proposed to avoid the initialization problem and thus lead to an efficient solution; the anchor nodes are closely related to the base nodes in [18].

Map joining has shown to be able to improve the efficiency for large-scale SLAM as well as reduce the linearisation errors. Similar to recursive nonlinear estimation (e.g. [7]), map joining also solves a recursive version of the problem step by step, thus rarely gets stuck in local minima. However, when the local map information is summarized as local map estimate and its covariance matrix, the map joining problem is slightly different from the original full nonlinear optimization SLAM problem using the original SLAM data. In general, the more accurate the local maps are, the closer the map joining result to the best solution to the full nonlinear least squares SLAM. Recently, good progress has been made in using nonlinear optimization techniques to solve small-scale SLAM with mathematical guarantees, such as one-step or two-step SLAM [25] or through verifications techniques [27]. These make the application of map joining techniques more promising.

Existing map joining algorithms are based on nonlinear estimation or nonlinear optimization, with no convergence guarantee. This paper proposes a new map joining algorithm, Linear SLAM, where only solving linear least squares and performing nonlinear coordinate transformations are needed, thus guarantees convergence. Although in Map joining 1.0 [13], the idea of first transforming one of the map using an estimate of relative pose from another map to have both submaps in a common reference is similar to Linear SLAM in the case of point features, significant inaccuracy is introduced in Map joining 1.0 because external relative pose information is used to perform the map transformation. This can be clearly seen from its poor performance as compared to Map joining 2.0 [12]. While, Linear SLAM has shown to perform better than most of the existing nonlinear map joining algorithms (see Section 8).

The following sections will explain the details of Linear SLAM.

3 Different Local Submaps in SLAM

The aim of this section is to show that we have the freedom to choose the coordinate frame of the local and global maps, which is the fundamental idea of the proposed Linear SLAM and the reason why the traditional map joining methods are nonlinear without considering this. We also show that the marginalization of local maps makes the proposed Linear SLAM a unified linear approach for solving different SLAM problems.

A local submap in SLAM refers to a map built using only a small portion of the SLAM data. In a feature-based SLAM, the data used for building a local submap contains odometry and observation information related to a small set of robot poses and features. Fig. 1(a) shows an example of odometry and observation information related to three poses $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ and three features $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$.

3.1 The Freedom of Choosing the Coordinate Frame

Consider the problem of building a local map by performing nonlinear least squares optimization based on the odometry and observation information given in the pose-feature graph shown in Fig. 1(a). Since the odometry and observation are all relative information, we must specify the coordinate frame when building a map.

One common choice in SLAM is to fix the first pose $\mathbf{P}_0$ as $(0, 0, 0)$ to define the coordinate frame, then we can obtain a local map shown in Fig. 1(b) after the nonlinear least squares optimization. The state vector of the local map contains two poses $\mathbf{P}_1$ and $\mathbf{P}_2$ and three features, all in the coordinate frame of $\mathbf{P}_0$.

Alternatively, we can fix the last pose $\mathbf{P}_2$ as $(0, 0, 0)$ to define the coordinate frame, then we can obtain a local map shown in Fig. 1(c). The state vector of the local map contains two poses $\mathbf{P}_0$ and $\mathbf{P}_1$ and three features, all in the coordinate frame of $\mathbf{P}_2$.

The choice of the coordinate frame can also be that defined by two features $\mathbf{F}_1$ and $\mathbf{F}_2$, that is, the feature position $\mathbf{F}_1$ is fixed as $(0, 0)$ and the x-axis is along the direction from $\mathbf{F}_1$ to $\mathbf{F}_2$. In this case, we can obtain a local map as shown in Fig. 1(d). The state vector of this local map contains three poses $\mathbf{P}_0, \mathbf{P}_1$ and $\mathbf{P}_2$ and feature $\mathbf{F}_3$ as well as the x-coordinate of feature $\mathbf{F}_2$, all in the coordinate frame defined by the two features $\mathbf{F}_1$ and $\mathbf{F}_2$.

The uncertainty of the local map is described by the associated covariance matrix (or information matrix). The above three local maps (including their covariance matrices) can be transferred from one to another by performing a coordinate transformation.

3.2 The Option of Marginalization

If one wants to reduce the size of the local map, marginalization can be performed after building the local map containing all the poses and features. For example, one can marginalize out all the poses apart from the last pose, then a state vector as that used in EKF SLAM is obtained [1]. One can also marginalize out all the features, then a state vector as that used in pose graph SLAM is obtained [9]. Alternatively, one can also marginalize out all the poses, then a state vector only containing features can be obtained, as that used in D-SLAM mapping [28].

3

(a) The odometry and observation information for building a local map

(b) Local map with coordinate frame defined by $\mathbf{P}_0$

(c) Local map with coordinate frame defined by $\mathbf{P}_2$

(d) Local map with coordinate frame defined by $\mathbf{F}_1$ and $\mathbf{F}_2$

Fig. 1. Different local maps built from the same odometry and observation information. Here $\mathbf{P}_i$ represents the robot pose and $\mathbf{F}_j$ represents the feature position.

After marginalization, the new covariance matrix (or information matrix) corresponding to the state vector with reduced dimension can be obtained.

In this paper, a map that contains both poses and features is called a pose-feature map; a map that only contains poses is called a pose-only map; while a map that only contains features is called a feature-only map.

3.3 The Idea of Map Joining

Assuming each local map estimate is consistent, map joining [14, 15] uses each local map as an integrated observation to replace the original odometry and observation information. This significantly reduces the computational cost for large-scale SLAM. The covariance matrices of the local maps play an important role since they describe the uncertainties of the integrated observations. Note that when computing the local map covariance matrix (information matrix), the covariance matrices of the original odometry and observation information are used, thus the local map can be regarded as a summary of the odometry and observation information involved.

The local maps need to have common poses or features such that they can be joined together. At least one com-

mon pose or two common features for 2D (three common features for 3D) are required to join two local maps together.

The key point of this section is that we have the freedom to choose the coordinate frame of a local map. Moreover, we also have the freedom to choose the coordinate frame of the global map. In this paper, we will demonstrate that by judiciously selecting the coordinate frames of the local maps at different map joining steps, the map joining result can be obtained by solving linear least squares problems and performing nonlinear coordinate transformations. This is the major difference to the traditional map joining. The proposed approach can be applied to the joining of different versions of local maps including pose-feature maps, pose-only maps and feature-only maps. Thus, a unified linear approach for solving SLAM problems is obtained. The following sections provide some details of the approach.

4 Joining Two Pose-feature Maps using Linear Least Squares

A pose-feature map consists of a number of features and either one robot pose (such as the map built by conventional EKF SLAM) or a number of robot poses (such as

4

Fig. 2. The proposed new way of building and joining two maps.

the map built by nonlinear least squares optimization without marginalizing out poses).

4.1 The Coordinate Frames of the Maps

In this paper, we propose to choose the coordinate frames of the two pose-feature maps and the combined map as follows (see Fig. 2).

Map 1 is built in the coordinate frame defined by its end pose $\mathbf{P}_1$. Map 2 is built in the coordinate frame defined by its start pose $\mathbf{P}_1$ (which is the same as the end pose of Map 1) [1]. The coordinate frame of the combined map, Map 12, is defined by $\mathbf{P}_2$, the robot end pose of Map 2.

In the following, we will show that although the joining of Map 1 and Map 2 to get Map 12 in Fig. 2 is still a nonlinear optimization problem, it is equivalent to a linear least squares optimization problem plus a (nonlinear) coordinate transformation.

---

[1] Map 1 and Map 2 can either be small local maps built using the local odometry and observation information, or be larger maps as the result of combining a number of small local maps. There can be some other robot poses in Map 1 and Map 2, they are not shown in Fig. 2 for simplification.

4.2 Nonlinear Least Squares Formulation of Joining Two Maps

Suppose Map 1 and Map 2 in Fig. 2 are denoted by M^{L_1} and M^{L_2} and are given by

$$M^{L_1} = (\hat{\mathbf{X}}^{L_1},\ I^{L_1}), \quad M^{L_2} = (\hat{\mathbf{X}}^{L_2},\ I^{L_2}) \qquad (1)$$

where $\hat{\mathbf{X}}^{L_1}$ and $\hat{\mathbf{X}}^{L_2}$ are the estimates of the state vectors $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ of each map, and I^{L_1} and I^{L_2} are the associated information matrices.

The state vectors $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ are defined as [2]

$$\begin{aligned} \mathbf{X}^{L_1} &= [\mathbf{P}_0^{L_1},\ \mathbf{X}_{F_1}^{L_1},\ \mathbf{X}_{F_{12}}^{L_1}] \\ \mathbf{X}^{L_2} &= [\mathbf{P}_2^{L_2},\ \mathbf{X}_{F_2}^{L_2},\ \mathbf{X}_{F_{12}}^{L_2}]. \end{aligned} \qquad (2)$$

Note that M^{L_1} is in the coordinate frame of $\mathbf{P}_1$, and M^{L_2} is also in the coordinate frame of $\mathbf{P}_1$. Here $\mathbf{P}_1$ is the end pose of M^{L_1} (also the start pose of M^{L_2}). It defines the coordinate frame of both M^{L_1} and M^{L_2} and thus is not included in the state vectors of the two maps.

In the state vectors $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ in (2), $\mathbf{X}_{F_1}^{L_1}$ and $\mathbf{X}_{F_2}^{L_2}$ represent the features only appear in M^{L_1} or M^{L_2}, respectively, while $\mathbf{X}_{F_{12}}^{L_1}$ and $\mathbf{X}_{F_{12}}^{L_2}$ represent the common features appear in both the two maps.

In the state vector $\mathbf{X}^{L_1}$ in (2), the start pose $\mathbf{P}_0$ of M^{L_1} is presented by

$$\mathbf{P}_0^{L_1} = [\mathbf{t}_0^{L_1},\ \mathbf{r}_0^{L_1}] \qquad (3)$$

where $\mathbf{t}_0^{L_1}$ is the translation vector, and $\mathbf{r}_0^{L_1}$ is/are the rotation angle/angles of pose $\mathbf{P}_0^{L_1}$. Similarly, in the state vector $\mathbf{X}^{L_2}$ in (2), the end pose $\mathbf{P}_2$ of M^{L_2} is presented by

$$\mathbf{P}_2^{L_2} = [\mathbf{t}_2^{L_2},\ \mathbf{r}_2^{L_2}]. \qquad (4)$$

Now the state estimates $\hat{\mathbf{X}}^{L_1}$ and $\hat{\mathbf{X}}^{L_2}$ can be written clearly as

$$\begin{aligned} \hat{\mathbf{X}}^{L_1} &= [\hat{\mathbf{t}}_0^{L_1},\ \hat{\mathbf{r}}_0^{L_1},\ \hat{\mathbf{X}}_{F_1}^{L_1},\ \hat{\mathbf{X}}_{F_{12}}^{L_1}] \\ \hat{\mathbf{X}}^{L_2} &= [\hat{\mathbf{t}}_2^{L_2},\ \hat{\mathbf{r}}_2^{L_2},\ \hat{\mathbf{X}}_{F_2}^{L_2},\ \hat{\mathbf{X}}_{F_{12}}^{L_2}]. \end{aligned} \qquad (5)$$

Our goal is to join M^{L_1} and M^{L_2} to get the map $M^{G_{12}}$, where $M^{G_{12}}$ is in the coordinate frame of $\mathbf{P}_2$.

---

[2] To simplify the notations, the 'transpose's in the state vectors are sometimes omitted in this paper. For example, $\mathbf{X}^{L_1}, \mathbf{P}_0^{L_1}, \mathbf{X}_{F_1}^{L_1}, \mathbf{X}_{F_{12}}^{L_1}$ are all column vectors and the rigorous notation should be $\mathbf{X}^{L_1} = [(\mathbf{P}_0^{L_1})^T,\ (\mathbf{X}_{F_1}^{L_1})^T,\ (\mathbf{X}_{F_{12}}^{L_1})^T]^T$.

Fig. 3. Linear way to build Map 12.

The state vector of $M^{G_{12}}$ containing pose $\mathbf{P}_0$, pose $\mathbf{P}_1$ and all the features is defined as

$$
\begin{aligned}
\mathbf{X}^{G_{12}} &= [\mathbf{P}_0^{G_{12}},\ \mathbf{P}_1^{G_{12}},\ \mathbf{X}_{F_1}^{G_{12}},\ \mathbf{X}_{F_2}^{G_{12}},\ \mathbf{X}_{F_{12}}^{G_{12}}] \\
&= [\mathbf{t}_0^{G_{12}},\ \mathbf{r}_0^{G_{12}},\ \mathbf{t}_1^{G_{12}},\ \mathbf{r}_1^{G_{12}},\ \mathbf{X}_{F_1}^{G_{12}},\ \mathbf{X}_{F_2}^{G_{12}},\ \mathbf{X}_{F_{12}}^{G_{12}}]
\end{aligned}
\tag{6}
$$

where all the variables in the state vector $\mathbf{X}^{G_{12}}$ are in the coordinate frame of $\mathbf{P}_2$.

Similar to [19], in the map joining problem, the state estimates $\hat{\mathbf{X}}^{L_1}$ and $\hat{\mathbf{X}}^{L_2}$ are treated as two integrated "observations" where the associated information matrices I^{L_1} and I^{L_2} describe the uncertainties of the observations. The optimization problem of joining the two maps is to minimize the following objective function

$$
f(\mathbf{X}^{G_{12}}) = \|\mathbf{e}_1\|_{I^{L_1}}^2 + \|\mathbf{e}_2\|_{I^{L_2}}^2
$$

$$
=
\left\|
\begin{bmatrix}
R_1\,(\mathbf{t}_0^{G_{12}} - \mathbf{t}_1^{G_{12}}) - \hat{\mathbf{t}}_0^{L_1} \\
r^{-1}(R_0 R_1^T) - \hat{\mathbf{r}}_0^{L_1} \\
R_1\,(\mathbf{X}_{F_1}^{G_{12}} - \mathbf{t}_1^{G_{12}}) - \hat{\mathbf{X}}_{F_1}^{L_1} \\
R_1\,(\mathbf{X}_{F_{12}}^{G_{12}} - \mathbf{t}_1^{G_{12}}) - \hat{\mathbf{X}}_{F_{12}}^{L_1}
\end{bmatrix}
\right\|_{I^{L_1}}^2
\tag{7}
$$

$$
+
\left\|
\begin{bmatrix}
-R_1\,\mathbf{t}_1^{G_{12}} - \hat{\mathbf{t}}_2^{L_2} \\
r^{-1}(R_1^T) - \hat{\mathbf{r}}_2^{L_2} \\
R_1\,(\mathbf{X}_{F_2}^{G_{12}} - \mathbf{t}_1^{G_{12}}) - \hat{\mathbf{X}}_{F_2}^{L_2} \\
R_1\,(\mathbf{X}_{F_{12}}^{G_{12}} - \mathbf{t}_1^{G_{12}}) - \hat{\mathbf{X}}_{F_{12}}^{L_2}
\end{bmatrix}
\right\|_{I^{L_2}}^2
$$

where $\mathbf{e}_i, (i = 1, 2)$ is the residual vector of the integrated "observation", and $\|\mathbf{e}_i\|_{I^{L_i}}^2 = \mathbf{e}_i^T I^{L_i} \mathbf{e}_i$ $(i = 1, 2)$ denotes the weighted norm of vector $\mathbf{e}_i$ with the given information matrix I^{L_i}.

In (7), $R_0 = r(\mathbf{r}_0^{G_{12}})$ and $R_1 = r(\mathbf{r}_1^{G_{12}})$ are the rotation matrices of pose $\mathbf{P}_0^{G_{12}}$ and $\mathbf{P}_1^{G_{12}}$ in the global state vector $\mathbf{X}^{G_{12}}$, respectively. And $r(\cdot)$ and $r^{-1}(\cdot)$ are the angle-to-

matrix and matrix-to-angle functions. For 2D scenarios, $r^{-1}(R_0 R_1^T) = \mathbf{r}_0^{G_{12}} - \mathbf{r}_1^{G_{12}}$ and $r^{-1}(R_1^T) = -\mathbf{r}_1^{G_{12}}$.

The problem of minimizing (7) is a nonlinear least squares problem. Solving it we can obtain the estimate of Map 12 together with its information matrix as

$$
M^{G_{12}} = (\hat{\mathbf{X}}^{G_{12}},\ I^{G_{12}}).
\tag{8}
$$

4.3 Linear Method to Obtain $M^{G_{12}}$

If we define new variables as follows (they are actually the seven distinct (nonlinear) functions in (7))

$$
\begin{aligned}
\bar{\mathbf{t}}_0^{G_{12}} &= R_1\,(\mathbf{t}_0^{G_{12}} - \mathbf{t}_1^{G_{12}}) \\
\bar{\mathbf{r}}_0^{G_{12}} &= r^{-1}(R_0 R_1^T) \\
\bar{\mathbf{t}}_2^{G_{12}} &= -R_1\,\mathbf{t}_1^{G_{12}} \\
\bar{\mathbf{r}}_2^{G_{12}} &= r^{-1}(R_1^T) \\
\bar{\mathbf{X}}_{F_1}^{G_{12}} &= R_1\,(\mathbf{X}_{F_1}^{G_{12}} - \mathbf{t}_1^{G_{12}}) \\
\bar{\mathbf{X}}_{F_2}^{G_{12}} &= R_1\,(\mathbf{X}_{F_2}^{G_{12}} - \mathbf{t}_1^{G_{12}}) \\
\bar{\mathbf{X}}_{F_{12}}^{G_{12}} &= R_1\,(\mathbf{X}_{F_{12}}^{G_{12}} - \mathbf{t}_1^{G_{12}})
\end{aligned}
\tag{9}
$$

and define a new state vector as

$$
\begin{aligned}
\bar{\mathbf{X}}^{G_{12}} &= [\bar{\mathbf{t}}_0^{G_{12}},\ \bar{\mathbf{r}}_0^{G_{12}},\ \bar{\mathbf{t}}_2^{G_{12}},\ \bar{\mathbf{r}}_2^{G_{12}},\ \bar{\mathbf{X}}_{F_1}^{G_{12}},\ \bar{\mathbf{X}}_{F_2}^{G_{12}},\ \bar{\mathbf{X}}_{F_{12}}^{G_{12}}] \\
&= g(\mathbf{X}^{G_{12}})
\end{aligned}
\tag{10}
$$

then the nonlinear least squares problem to minimize the objective function (7) becomes a linear least squares problem to minimize the following objective function

$$
\bar{f}(\bar{\mathbf{X}}^{G_{12}}) =
\left\|
\begin{bmatrix}
\bar{\mathbf{t}}_0^{G_{12}} - \hat{\mathbf{t}}_0^{L_1} \\
\bar{\mathbf{r}}_0^{G_{12}} - \hat{\mathbf{r}}_0^{L_1} \\
\bar{\mathbf{X}}_{F_1}^{G_{12}} - \hat{\mathbf{X}}_{F_1}^{L_1} \\
\bar{\mathbf{X}}_{F_{12}}^{G_{12}} - \hat{\mathbf{X}}_{F_{12}}^{L_1}
\end{bmatrix}
\right\|_{I^{L_1}}^2
+
\left\|
\begin{bmatrix}
\bar{\mathbf{t}}_2^{G_{12}} - \hat{\mathbf{t}}_2^{L_2} \\
\bar{\mathbf{r}}_2^{G_{12}} - \hat{\mathbf{r}}_2^{L_2} \\
\bar{\mathbf{X}}_{F_2}^{G_{12}} - \hat{\mathbf{X}}_{F_2}^{L_2} \\
\bar{\mathbf{X}}_{F_{12}}^{G_{12}} - \hat{\mathbf{X}}_{F_{12}}^{L_2}
\end{bmatrix}
\right\|_{I^{L_2}}^2 .
\tag{11}
$$

This linear least squares problem can be written in a compact form as

$$
\text{minimize } \bar{f}(\bar{\mathbf{X}}^{G_{12}}) = \|A\,\bar{\mathbf{X}}^{G_{12}} - \mathbf{Z}\|_{I_Z}^2
\tag{12}
$$

where $\mathbf{Z}$ is the constant vector combining the state estimates of the two maps

$$
\mathbf{Z} = [\hat{\mathbf{X}}^{L_1},\ \hat{\mathbf{X}}^{L_2}],
\tag{13}
$$

I_Z is the corresponding information matrix given by

$$
I_Z = \mathrm{diag}(I^{L_1}, I^{L_2}),
\tag{14}
$$

$\bar{\mathbf{X}}^{G_{12}}$ is the state vector represents the global map defined in (10). The coefficient matrix A is sparse and is given by

$$A = \begin{bmatrix} E & & & & & & & \\ & E & & & & & & \\ & & E & & & & & \\ & & & E & & & & \\ & E & & & & & & \\ & & E & & & & & \\ & & & E & & & & \\ & & & & E \end{bmatrix} \quad (15)$$

where E denotes the identity matrix with the size corresponding to the different variables in the state vector $\bar{\mathbf{X}}^{G_{12}}$.

The optimal solution $\hat{\bar{\mathbf{X}}}^{G_{12}}$ to the linear least squares problem (12) can be obtained by solving the sparse linear equation

$$A^T I_Z A \, \hat{\bar{\mathbf{X}}}^{G_{12}} = A^T I_Z \mathbf{Z}. \quad (16)$$

The corresponding information matrix can be computed by

$$\bar{I}^{G_{12}} = A^T I_Z A. \quad (17)$$

From (10), we have $\mathbf{X}^{G_{12}} = g^{-1}(\bar{\mathbf{X}}^{G_{12}})$. Note that the function $g^{-1}(\cdot)$ has a closed-form as follows:

$$\mathbf{X}^{G_{12}} = g^{-1}(\bar{\mathbf{X}}^{G_{12}})$$
$$\Rightarrow \begin{cases} \mathbf{t}_0^{G_{12}} = \bar{R}_2 \, (\bar{\mathbf{t}}_0^{G_{12}} - \bar{\mathbf{t}}_2^{G_{12}}) \\ \mathbf{r}_0^{G_{12}} = r^{-1}(\bar{R}_0 \bar{R}_2^T) \\ \mathbf{t}_1^{G_{12}} = -\bar{R}_2 \, \bar{\mathbf{t}}_2^{G_{12}} \\ \mathbf{r}_1^{G_{12}} = r^{-1}(\bar{R}_2^T) \\ \mathbf{X}_{F_1}^{G_{12}} = \bar{R}_2 \, (\bar{\mathbf{X}}_{F_1}^{G_{12}} - \bar{\mathbf{t}}_2^{G_{12}}) \\ \mathbf{X}_{F_2}^{G_{12}} = \bar{R}_2 \, (\bar{\mathbf{X}}_{F_2}^{G_{12}} - \bar{\mathbf{t}}_2^{G_{12}}) \\ \mathbf{X}_{F_{12}}^{G_{12}} = \bar{R}_2 \, (\bar{\mathbf{X}}_{F_{12}}^{G_{12}} - \bar{\mathbf{t}}_2^{G_{12}}) \end{cases} \quad (18)$$

where $\bar{R}_2 = r(\bar{\mathbf{r}}_2^{G_{12}})$, $\bar{R}_0 = r(\bar{\mathbf{r}}_0^{G_{12}})$ are the rotation matrices of pose $\mathbf{P}_2$ and pose $\mathbf{P}_0$.

Now the solution to the nonlinear least squares problem (7) can be obtained by

$$\hat{\mathbf{X}}^{G_{12}} = g^{-1}(\hat{\bar{\mathbf{X}}}^{G_{12}}). \quad (19)$$

The corresponding information matrix $I^{G_{12}}$ can be obtained by

$$I^{G_{12}} = \nabla^T \bar{I}^{G_{12}} \nabla \quad (20)$$

where ∇ is the Jacobian of $\bar{\mathbf{X}}^{G_{12}}$ with respect to $\mathbf{X}^{G_{12}}$, evaluated at $\hat{\mathbf{X}}^{G_{12}}$

$$\nabla = \frac{\partial g(\mathbf{X}^{G_{12}})}{\partial \mathbf{X}^{G_{12}}} \big|_{\hat{\mathbf{X}}^{G_{12}}}. \quad (21)$$

Remark 1. An intuitive explanation of the above process is shown in Fig. 3. In fact, $\bar{\mathbf{X}}^{G_{12}} = g(\mathbf{X}^{G_{12}})$ in (9) and (10) is the coordinate transformation function, transforming pose $\mathbf{P}_0$, $\mathbf{P}_1$ and feature $\mathbf{F}$ in the coordinate frame of $\mathbf{P}_2$, into $\mathbf{P}_0$, $\mathbf{P}_2$ and $\mathbf{F}$ in the coordinate frame of $\mathbf{P}_1$. Thus $\mathbf{X}^{G_{12}} = g^{-1}(\bar{\mathbf{X}}^{G_{12}})$ in (18) has the closed-form formula which is another coordinate transformation, from $\mathbf{P}_0$, $\mathbf{P}_2$ and $\mathbf{F}$ in the coordinate frame of $\mathbf{P}_1$, back to $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{F}$ in the coordinate frame of $\mathbf{P}_2$. The linear way to solve the map joining problem is equivalent to solving a linear least squares problem first and then performing a coordinate transformation, as illustrated in Fig. 3. The fact that the two maps are built in the same coordinate frame is the key to make this linear map joining approach possible.

The following lemma states rigorously that the map $M^{G_{12}}$ obtained using the linear method is equivalent to that obtained using the nonlinear method.

Lemma 1: Joining M^{L_1} and M^{L_2} to build the global map $M^{G_{12}}$ in the coordinate frame of the end pose of M^{L_2}, is equivalent to first building the global map $\bar{M}^{G_{12}}$ in the coordinate frame of the start pose of M^{L_2}, and then transforming the global map $\bar{M}^{G_{12}}$ into $M^{G_{12}}$ by coordinate transformation.

Proof: See Appendix A.

4.4 Traditional Way of Building and Joining Two Maps

As comparison, the basic step in traditional map joining (such as sequential [14,15] or Divide and Conquer [12]) is also joining two maps, the process of which is illustrated in Fig. 4.

In traditional map joining, Map 1 is built in the coordinate frame defined by its start pose $\mathbf{P}_0$. Map 2 is built in the coordinate frame defined by its start pose $\mathbf{P}_1$ (the end pose of Map 1). Comparing to (1) and (5), Map 1 in traditional map joining is

$$M^{L_1^0} = (\hat{\mathbf{X}}^{L_1^0}, \ I^{L_1^0})$$
$$\hat{\mathbf{X}}^{L_1^0} = [\hat{\mathbf{t}}_1^{L_1^0}, \ \hat{\mathbf{r}}_1^{L_1^0}, \ \hat{\mathbf{X}}_{F_1}^{L_1^0}, \ \hat{\mathbf{X}}_{F_{12}}^{L_1^0}] \quad (22)$$

where $\mathbf{P}_0$ defines the coordinate frame of $M^{L_1^0}$, and thus is not included in the state vector of the Map 1. Map 2 M^{L_2} is the same as used in the Linear SLAM in Section 4.2.

Fig. 4. Traditional way of joining two maps (nonlinear, e.g. in SLSJF [15]).

The coordinate frame of the combined map is defined by $\mathbf{P}_0$, which is the same as Map 1 $M^{L_1^0}$. The state vector of the combined map in traditional map joining is

$$\mathbf{X}^{G_{12}^0} = [\mathbf{t}_1^{G_{12}^0},\ \mathbf{r}_1^{G_{12}^0},\ \mathbf{t}_2^{G_{12}^0},\ \mathbf{r}_2^{G_{12}^0},\ \mathbf{X}_{F_1}^{G_{12}^0},\ \mathbf{X}_{F_2}^{G_{12}^0},\ \mathbf{X}_{F_{12}}^{G_{12}^0}]. \tag{23}$$

Joining these two maps optimally is clearly a nonlinear optimization problem [19] because the coordinate frames of the two maps are different

$$\tilde{f}(\mathbf{X}^{G_{12}^0}) =$$

$$\left\| \begin{bmatrix} \mathbf{t}_1^{G_{12}^0} - \hat{\mathbf{t}}_1^{L_1^0} \\ \mathbf{r}_1^{G_{12}^0} - \hat{\mathbf{r}}_1^{L_1^0} \\ \mathbf{X}_{F_1}^{G_{12}^0} - \hat{\mathbf{X}}_{F_1}^{L_1^0} \\ \mathbf{X}_{F_{12}}^{G_{12}^0} - \hat{\mathbf{X}}_{F_{12}}^{L_1^0} \end{bmatrix} \right\|_{I^{L_1^0}}^2 + \left\| \begin{bmatrix} R_1^0\,(\mathbf{t}_2^{G_{12}^0} - \mathbf{t}_1^{G_{12}^0}) - \hat{\mathbf{t}}_2^{L_2} \\ r^{-1}(R_2^0\,R_1^{0^T}) - \hat{\mathbf{r}}_2^{L_2} \\ R_1^0\,(\mathbf{X}_{F_2}^{G_{12}^0} - \mathbf{t}_1^{G_{12}^0}) - \hat{\mathbf{X}}_{F_2}^{L_2} \\ R_1^0\,(\mathbf{X}_{F_{12}}^{G_{12}^0} - \mathbf{t}_1^{G_{12}^0}) - \hat{\mathbf{X}}_{F_{12}}^{L_2} \end{bmatrix} \right\|_{I^{L_2}}^2 \tag{24}$$

where $R_1^0 = r(\mathbf{r}_1^{G_{12}^0})$ and $R_2^0 = r(\mathbf{r}_2^{G_{12}^0})$ are the rotation matrices of poses $\mathbf{P}_1$ and $\mathbf{P}_2$ in the global state vector $\mathbf{X}^{G_{12}^0}$, respectively.

Thus, in the traditional map joining algorithms, the vector in the first term of objective function (24) is linear, while the vector in the second term in (24) is nonlinear. And it is impossible to choose an intermediate state vector to make (24) linear.

5 Joining A Sequence of Local Maps

Based on the linear solution to joining two maps as shown in Fig. 2, Fig. 3 and Section 4.3, we can use either sequential map joining or Divide and Conquer map joining to solve large-scale SLAM problems.

5.1 Proposed Sequential Map Joining

The new sequential map joining process we proposed can be illustrated in Fig. 5(a). Please note: (1) Local map 1 is built in the coordinate frame of its end pose instead of its start pose; Thus it can be fused with Local map 2 using the linear method in Section 4.3. (2) Global map 12, the result of joining Local map 1 and Local map 2, is in the coordinate frame of the end pose of Local map 2. Thus it can be fused with Local map 3 using the linear method in Section 4.3.

As comparison, in traditional map joining algorithms (such as [14, 15]), all the initial local map 1, 2 and 3 are built in the coordinate frames of their start poses. And at each step, the global map (map 12 or map 123) is always built in the coordinate frame of its start pose which is the coordinate frame of map 1.

5.2 Proposed Divide and Conquer Map Joining

The new Divide and Conquer map joining process we proposed is illustrated in Fig. 5(b). Please note: (1) Local map 1 and Local map 2 are built in the same coordinate frame; Local map 3 and Local map 4 are built in the same coordinate frame; (2) Global map 12 is built in the same coordinate frame as Global map 34. Thus Global map 12 and Global map 34 can be joined together using the linear method in Section 4.3.

While in traditional map joining (such as [12]), local map 1, 2, 3 and 4 are all in the coordinate frames of their start poses. Global map 12 is built in the same coordinate frame as local map 1. Global map 34 is built in the same coordinate frame as local map 3. And the final global map 1234 is built in the same coordinate frame as local map 1.

Remark 2. Local map 1 in Fig. 5(a) or Fig. 5(b) can be simply built by performing a nonlinear least squares using all the observation and odometry data with state vector defined as the robot poses and feature positions in the coordinate frame of the robot end pose in the local map (see Fig. 1(c)). Alternatively, we can first build the local map in the coordinate frame of the robot start pose $\mathbf{P}_0$, and then apply a coordinate transformation.

5.3 Computational Complexity

Suppose in the feature-based SLAM problem, the total number of feature/odometry observations is O_G and the

(a) Proposed sequential map joining.

(b) Proposed Divide and Conquer map joining.

Fig. 5. Sequential map joining and Divide and Conquer map joining.

number of robot poses/features in the state vector is S_G. In each iteration, the nonlinear system is first linearized and the linear least squares to be solved in each iteration is similar to the problem in (12). Then, the computational complexity of one iteration for the full nonlinear least squares SLAM is $\Theta(O_G + S_G^3)$, where the computational complexity of computing the information matrix $J^T I_Z J$ in (16) (here A is replaced by J, where J is the Jacobian of the observations w.r.t the state vector) is proportional to the number of the observations $\Theta(O_G)$ [48], and $\Theta(S_G^3)$ is the computational complexity of solving the traditional resolution of the linear system (16) [48] [3] . Thus, by considering the iteration number m, the computational complexity of solving nonlinear least squares SLAM is $\Theta(mO_G + mS_G^3)$.

For the proposed Linear SLAM algorithm, the computational cost consists of two parts: building the local maps and joining the local maps.

Suppose the observations and the robot poses/features in the state vector are equally divided to build n local maps. Thus in each local map, there are $\frac{1}{n}O_G$ observations and $\frac{1}{n}S_G$ poses and features. Similar to building the global map described above, if each local map is built

------

[3] The sparsity and ordering of the information matrix also have a significant impact on the computational cost of solving linear system. This is not discussed here.

by using nonlinear least squares SLAM with m iterations, the computational complexity of building n local maps is $\Theta(\frac{1}{n}O_G + (\frac{1}{n}S_G)^3) \times m \times n = \Theta(mO_G + \frac{m}{n^2}S_G^3)$. Thus, the computational complexity of building n local maps is much less than that of directly building the global map by nonlinear least squares SLAM.

For the linear joining of two local maps in Section 4, the computational complexity of computing the information matrix in (12) depends on the number of observations in the two local maps, which is $\Theta(\frac{2}{n}S_G)$. While, the computational complexities of solving the linear system (16) and transforming the joint map in (19) and (20) both depends on the number of poses/features in the state vector of the joint map, which are $\Theta((\frac{2}{n}S_G)^3)$ and $\Theta(\frac{2}{n}S_G)$, respectively. Thus, the computational complexity of joining of two local maps using the proposed linear method is $\Theta(\frac{4}{n}S_G + \frac{8}{n^3}S_G^3)$. The iteration number m is not applied here, since it is linear least squares optimization.

Thus, when joining a sequence of local maps in the sequential way, the overall computational complexity of the linear SLAM algorithm is

$$\Theta(mO_G + \frac{m}{n^2}S_G^3) + \sum_{i=2}^{n} \Theta(\frac{2i}{n}S_G + (\frac{i}{n}S_G)^3), \quad (25)$$

where i is the ID of the current local map to be joined.

9

Table 1
Computational Complexity† of Linear SLAM and Nonlinear Map Joining with Different Local Map Numbers (n) in Comparison with Nonlinear Least Squares SLAM
Victoria Park dataset, $O_G = 52288$, $S_G = 7197$ and suppose $m = 10$.

| | | Linear SLAM | | | | Nonlinear Map Joining | | | |
| | | Sequential | | Divide and Conquer | | Sequential | | Divide and Conquer | |
n	Local Map	Joining	Overall	Joining	Overall	Joining	Overall	Joining	Overall
1*	1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	0.0400	0.1792	0.2192	0.1640	0.2040	1.7920	1.8320	1.6400	1.6800
10	0.0100	0.3024	0.3124	0.1680	0.1780	3.0240	3.0340	1.6800	1.6900
20	0.0025	0.5512	0.5537	0.1690	0.1715	5.5124	5.5149	1.6900	1.6925
50	4.0e-4	1.3005	1.3009	0.1393	0.1397	13.005	13.005	1.3928	1.3932
100	1.0e-4	2.5502	2.5503	0.1393	0.1394	25.502	25.503	1.3932	1.3933

†The computational complexity is relative to the nonlinear least squares SLAM.
*One local map means it is the nonlinear least squares SLAM to obtain the global map directly. The computational complexity is set to 1 as the benchmark for comparison.

While, in the Divide and Conquer manner, the overall computational complexity of the linear SLAM algorithm is

$$\Theta(mO_G + \frac{m}{n^2}S_G^3) + \Theta(2S_G + S_G^3)$$
$$+ \sum_{k=1}^{c(\log_2(n))-1} \Theta(\frac{2^{k+1}}{n}S_G + (\frac{2^k}{n}S_G)^3) \times z(\frac{n}{2^k}) \quad (26)$$

where k is the level in the Divide and Conquer process, $c(\cdot)$ is the function which round toward positive infinity and $z(\cdot)$ is the function which round toward zero. Thus, $c(\log_2(n))$ is the number of levels and $z(\frac{n}{2^k})$ is the number of times of performing joining two local maps at the kth level in the Divide and Conquer process.

In comparison, for the traditional nonlinear map joining algorithm using nonlinear least squares optimization, suppose the iteration number is also m when joining each of two local maps, the computation cost in the sequential manner is

$$\Theta(mO_G + \frac{m}{n^2}S_G^3) + \sum_{i=2}^{n} \Theta(\frac{mi}{n}S_G + m(\frac{i}{n}S_G)^3). \quad (27)$$

And the computation cost of the traditional nonlinear map joining algorithms in the Divide and Conquer manner is

$$\Theta(mO_G + \frac{m}{n^2}S_G^3) + \Theta(mS_G + mS_G^3)$$
$$+ \sum_{k=1}^{c(\log_2(n))-1} \Theta(\frac{2^k m}{n}S_G + m(\frac{2^k}{n}S_G)^3) \times z(\frac{n}{2^k}). \quad (28)$$

The computational complexity of the proposed Linear SLAM algorithm and the traditional map joining algorithm using nonlinear least squares optimization with respect to the global nonlinear least squares SLAM are shown in Table 1. As an example of using Victoria Park dataset, performing Linear SLAM in the Divide and Conquer manner the computational complexity is always much less than that of global nonlinear least squares SLAM. While, in the sequential manner the computational complexity can be more if the number of local maps is large. The number and size of local maps also affect the computational complexity of the map joining algorithm. Ideally as shown in Table 1, the larger the number of local maps are built from the original dataset, the more the computational complexity is in the sequential manner, while the computational complexity will remain similar in the Divide and Conquer manner. While, as shown in Fig. 8, the real computational cost also depends on the implementation of the algorithm, e.g. the memory operation in the implementation of Divide and Conquer.

6 Joining Two Pose-only Maps

Recently pose graph SLAM becomes popular where relative poses among the robot poses are computed using the sensor data and then an optimization is performed to obtain an estimate of the state vector containing only robot poses. The linear approach proposed in Section 4 and Section 5 can also be applied to the joining of pose-only maps. A pose-only local map can be the result of solving a small pose graph SLAM or the marginalization (of all the features) from a pose-feature local map. In this section, we explain the process of joining two pose-only maps using linear least squares, which can be illustrated using Fig. 2 (by replacing all the features by poses).

6.1 Local Pose-only Maps

Suppose there are two pose-only maps M^{L_1} and M^{L_2} given in the form of (1) where $\hat{\mathbf{X}}^{L_1}$ and $\hat{\mathbf{X}}^{L_2}$ are the estimates of the state vectors $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ defined as

$$\begin{aligned} \mathbf{X}^{L_1} &= (\mathbf{X}^{L_1}_{P_{L1}}, \ \mathbf{X}^{L_1}_{P_{L12}}) \\ \mathbf{X}^{L_2} &= (\mathbf{X}^{L_2}_{P_{L2}}, \ \mathbf{X}^{L_2}_{P_{L12}}) \end{aligned} \tag{29}$$

and I^{L_1} and I^{L_2} are the associated information matrices. Both M^{L_1} and M^{L_2} are in the coordinate frame of $\mathbf{P}_1$.

In the state vector $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ in (29), $\mathbf{X}^{L_1}_{P_{L12}}$ and $\mathbf{X}^{L_2}_{P_{L12}}$ are the common poses between M^{L_1} and M^{L_2}, while $\mathbf{X}^{L_1}_{P_{L1}}$ and $\mathbf{X}^{L_2}_{P_{L2}}$ are the poses that appear in only one of the maps. All the poses can be defined similarly as (3) in Section 4.2.

6.2 Joining Two Pose-only Maps by Linear Least Squares

Joining these two pose-only maps M^{L_1} and M^{L_2} to build the intermediate map $\bar{M}^{G_{12}} = (\hat{\bar{\mathbf{X}}}^{G_{12}}, \ \bar{I}^{G_{12}})$ can also be solved by linear least squares. Here the global state vector $\bar{\mathbf{X}}^{G_{12}}$ is defined as

$$\bar{\mathbf{X}}^{G_{12}} = (\bar{\mathbf{X}}^{G_{12}}_{P_{L1}}, \ \bar{\mathbf{X}}^{G_{12}}_{P_{L2}}, \ \bar{\mathbf{X}}^{G_{12}}_{P_{L12}}) \tag{30}$$

where $\bar{\mathbf{X}}^{G_{12}}_{P_{L12}}$ are the common poses, $\bar{\mathbf{X}}^{G_{12}}_{P_{L1}}$ and $\bar{\mathbf{X}}^{G_{12}}_{P_{L2}}$ are the poses that only appear in M^{L_1} and M^{L_2}, respectively. All the variables in the global state vector $\bar{\mathbf{X}}^{G_{12}}$ are in the coordinate frame of $\mathbf{P}_1$. This intermediate map $\bar{M}^{G_{12}}$ can then be transformed into $M^{G_{12}}$ by the nonlinear coordinate transformation, where $M^{G_{12}}$ is defined in the coordinate frame of $\mathbf{P}_2$.

Remark 3. It should be mentioned that, for the pose-feature map joining, the only common pose between the two local maps defines the coordinate frames of the two local maps as shown in Fig. 2; this pose is not in the state vectors of the two local maps. So there is no common pose in the observations $\mathbf{Z}$ in (13). While, for the pose-only map joining problem, there can be many common poses between two local maps, and the wraparound of the rotation angles of the common poses must be carefully considered when performing map joining. As pointed out in [29,30], wraparound of the orientation angles is a critical issue in the linear formulation. However, as only two maps are joined each time in the proposed Linear SLAM algorithm, it is sufficient to wrap the angle in one of the two observations of a common pose in $\mathbf{Z}$ to make the difference between the two angles in the observations fall into $(-\pi, \pi]$.

Fig. 6. The proposed way of building and joining two feature-only maps. Map 1 and Map 2 are both in the coordinate frame defined by Feature $\mathbf{F}_O$ and $\mathbf{F}_X$, while Map 12 is in the coordinate frame defined by Feature $\mathbf{F}'_O$ and $\mathbf{F}'_X$.

7 Joining Two Feature-only Maps

Apart from traditional feature-based SLAM and pose graph SLAM, an alternative approach for SLAM is decoupling the localization and mapping (D-SLAM [28]). In the D-SLAM mapping part, the map only consists of a number of features. In this paper, we call this kind of maps "feature-only maps". A feature-only map can also be obtained by marginalizing out all the poses from a pose-feature map (marginalizing out $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_2$ in the map in Fig. 1(d)). In this paper, we propose to use Linear SLAM to join 2D or 3D feature-only maps by first extending the idea in [28] and [31] into 3D case.

7.1 Local Feature-only Maps

Suppose there are two feature-only maps M^{L_1} and M^{L_2} given in the form of (1), where the state vectors $\mathbf{X}^{L_1}$ and $\mathbf{X}^{L_2}$ of each map are defined similarly to those in Section 6.1, by replacing the poses by features.

In feature-only maps, there is no robot pose in the map. Instead, two features for 2D [28] and three features for 3D are used to define the coordinate frame of the map. The details of the coordinates definition are described in Appendix B.

When joining two feature-only maps, at least two common features for 2D (three common features for 3D) are required to make sure there are enough information to combine the two maps together [31].

(a) VicPark200 (b) DLR200 (c) VicPark6898

(d) DLR3298 (e) 3D Simu871

Fig. 7. 2D and 3D pose-feature map joining results of Linear SLAM (LSLAM). Fig. 7(a) and Fig. 7(b) are the results of joining the 200 local maps for 2D Victoria Park and DLR datasets. Fig. 7(c) and Fig. 7(d) are the results of Victoria Park and DLR datasets by joining 6898/3298 local maps. The 2D results are compared with CF-SLAM, T-SAM, SLSJF and nonlinear least squares optimization. Fig. 7(e) is the result of a 3D simulation with 870 local maps. It is only compared to the ground truth as the algorithms compared above are only applicable for 2D case. The quantitive comparison is given in Table 2.

As shown in Fig. 6, for the proposed map joining approach, both M^{L_1} and M^{L_2} are in the same coordinate frame defined by the common features between them.

7.2 Joining Two Feature-only Maps by Linear Least Squares

The goal is to join the two feature-only maps M^{L_1} and M^{L_2} to build the feature-only map $M^{G_{12}} = (\hat{\mathbf{X}}^{G_{12}}, I^{G_{12}})$ as shown in Fig. 6. Similar to the joining of pose-feature maps, the intermediate map $\bar{M}^{G_{12}} = (\hat{\bar{\mathbf{X}}}^{G_{12}}, \bar{I}^{G_{12}})$ which is in the same coordinates as M^{L_1} and M^{L_2}, is first built by linear least squares, and then it is transformed into $M^{G_{12}}$.

Remark 4. The process of building $\bar{M}^{G_{12}}$ and the transformation from $\bar{M}^{G_{12}}$ to $M^{G_{12}}$ is similar to those described in Section 4.3. The transformation of the state vector of feature-only maps is different from that of pose-feature maps or pose-only maps since the coordinate frame is not defined by a pose. The details of the definition and the transformation of the coordinate frames in feature-only maps in both 2D and 3D cases are described in Appendix B.

8 Simulation and Experimental Results Using Benchmark Datasets

In this section, 2D and 3D simulation and real datasets of feature-based SLAM and pose graph SLAM have been used to evaluate the proposed Linear SLAM algorithm. All the Linear SLAM results presented below are from the Divide and Conquer map joining which is computationally less expensive (Section 5.3) than the sequential map joining. All the experiments are run on an Intel i5-3230M@2.6GHz CPU on a laptop.

8.1 Pose-feature Map Joining

8.1.1 2D Pose-feature Map Joining

For 2D pose-feature map joining, the Victoria Park dataset [32] and the DLR dataset [33] are used. Here, VicPark200(6898) / DLR200(3298) means the 200(6898) / 200(3298) local maps built from the Victoria Park / DLR dataset. These local map datasets are all available on OpenSLAM under project 2D-I-SLSJF.

The results of Linear SLAM using the four local map

12

Table 2
RMSE* and χ^2 of Different Feature-Based SLAM Algorithms

Dataset	CF-SLAM			T-SAM			SLSJF			Linear SLAM			Nonlinear LS
	Pose	Feature	χ^2	Pose	Feature	χ^2	Pose	Feature	χ^2	Pose	Feature	χ^2	χ^2
VicPark200	0.1949	0.3191	3654				0.2154	0.3599	**3644**	**0.0391**	**0.0643**	3645	3643
DLR200	0.2391	0.2400	34480				3.6574	3.6773	14603	**0.1537**	**0.1545**	**12577**	11164
VicPark6898	**0.0330**	**0.0509**	9098	0.9329	1.4080	117229	0.0718	0.1169	**9013**	0.1401	0.2348	9021	9013
DLR3298	0.6308	0.6339	85827	0.6064	0.5976	112474	2.5066	2.5449	**28316**	**0.0976**	**0.1042**	28373	27689

* RMSE (in meters) of the position of poses and features is w.r.t. the results of the full nonlinear least squares optimization (Nonlinear LS). For VicPark200 and DLR200 datasets, the Nonlinear LS means the optimization with all the local maps together as the observations. For VicPark6898 and DLR3298 datasets, the Nonlinear LS is the optimization with all the original observations. Because the building and joining submaps in T-SAM is different from the ones in the others, the RMSE and χ^2 of the results from T-SAM are w.r.t. the Nonlinear LS with the original observations.

(a) Computational Cost

(b) Accuracy

Fig. 8. The impact on the accuracy and computational cost by the number of local maps using the Victoria Park dataset.

datasets are shown in Fig. 7(a) to Fig. 7(d). The computational time are 1.42s, 0.47s, 2.83s and 0.85s, respectively. For comparison, the results from CF-SLAM [35], T-SAM [18], SLSJF [15] and nonlinear least squares optimization (Nonlinear LS) algorithms are also shown in the figures. The root-mean-square error (RMSE) of Linear SLAM and the other three algorithms as compared with the optimal nonlinear LS results are shown in Table 2. The final χ^2 errors of the different algorithms are also compared in Table 2.

For VicPark200 and DLR200 datasets, since the original inputs are the local maps (in the EKF SLAM format), we use I-SLSJF [19] (full nonlinear least squares with all the local maps as observations) as implementation for the Nonlinear LS. For VicPark6898 and DLR3298 datasets, since each local map was directly obtained using the observations of features from one pose together with the odometry to the next pose, the I-SLSJF results using these two datasets are equivalent to those using full least squares optimization with all the original observations and odometry information. Also for these two datasets, the whole Linear SLAM algorithm only requires linear least squares (plus nonlinear coordinate transformations) because there is no nonlinear optimization process for local map building involved.

For the comparison, in CF-SLAM, the inputs are also the local maps datasets from 2D-I-SLSJF on OpenSLAM, which are the same as the ones used in Linear SLAM and SLSJF. For T-SAM, because the building and joining of local maps are different from SLSJF, CF-SLAM and Linear SLAM, in our implementation, 200 local maps are built from the original observations and odometries for Victoria Park and DLR datasets in the way proposed in [18] (the way of cutting original dataset into local maps for T-SAM may not be optimal, and a better cut may further improve the results of T-SAM. However, it is not discussed in this paper). And as all the robot poses are kept in T-SAM, the accuracy and χ^2 of the results are w.r.t. the Nonlinear LS with the original observations.

As shown in Fig. 7(a) to Fig. 7(d) and Table 2, for the pose-feature map joining, in terms of RMSE, three out of four results of Linear SLAM are better than those of CF-SLAM and SLSJF, and all the four results are better than those of T-SAM, while nearly the same as the optimal Nonlinear LS results. The χ^2 of Linear SLAM results are smaller than those of CF-SLAM, similar to those from the SLSJF results, and very close to that of the optimal nonlinear least squares solutions.

8.1.2 Impact of the Size of the Local Maps

The impact of the size of local maps on the accuracy and computational cost by the proposed Linear SLAM algorithm is illustrated using the Victoria Park dataset. The result is shown in Fig. 8. The time used for building all the local maps decreases when the number of local maps increases from 5 to 100. For the map joining, the computational time slightly increases as the number of local maps increases due to the property of the hierarchical

(a) 2D Simu8240 dataset with 50 local maps

(b) 2D Simu35188 dataset with 700 local maps

Fig. 9. 2D feature-based SLAM simulations and Linear SLAM results for consistency check.

Divide and Conquer process. The total time of local map building plus map joining remains similar, in which the smallest is when the number of local maps is 20. In terms of the accuracy, the result does not vary much when the number of local maps is changed from 5 to 100. The RM-SEs of both the poses and features are the smallest when there are only 5 local maps. It should be noted that the impact of the size of local maps can be different for different datasets due to the different number of observations and the different number of poses/features.

8.1.3   Consistency Analysis

Two 2D simulation datasets Simu8240 (with 50 local maps) and Simu35188 (with 700 local maps) which are

Table 3
Consistency of Linear SLAM Results by NEES Check (95%)

Datasets	Dimension	95% bound	NEES
Simu8240	1224	1.3065e+03	1.2392e+03
Simu35188	8404	8.6184e+03	7.9163e+03

available on OpenSLAM (project 2D-I-SLSJF) are used to check the consistency of the proposed Linear SLAM algorithm for pose-feature map joining. Fig. 9(a) and Fig. 9(b) illustrate the simulation scenarios (the ground truth) and the results of pose-feature map joining using Linear SLAM.

All the features present in the final global map and the corresponding information matrix were used to compute the normalised estimation error squared (NEES) [36] for checking the consistency [37] of Linear SLAM algorithm. The results shown in Table 3 indicate that the Linear SLAM results are consistent.

The uncertainties obtained from the proposed Linear SLAM algorithm are also compared to the uncertainties obtained by performing the Nonlinear LS algorithm. Both VicPark6898 and DLR3298 datasets are employed to this comparative evaluation. The uncertainties of the features estimated from Linear SLAM and Nonlinear LS are shown in Fig. 10(a)(b) and Fig. 11(a)(b) in red and blue, respectively. And the uncertainties of the poses estimated from both two algorithms are shown in Fig. 10(c) and Fig. 11(c). As we can see from the figures, the uncertainties estimated from the proposed Linear SLAM algorithm is very close to those obtained from the Nonlinear LS algorithm, which indicates that the proposed Linear SLAM algorithm can obtain accurate results in term of both estimates and uncertainty.

8.1.4   3D Pose-feature Map Joining

For the 3D pose-feature map joining, a simulation is done with a trajectory of 871 poses and uniformly distributed features in the environment. The pose-feature map joining using Linear SLAM is applied with 870 local maps and the result is shown in Fig. 7(e). The RMSE of the position of poses and features by Linear SLAM as compared with the ground truth are 0.003266m and 0.003158m, respectively, with the computational cost of 1.00s.

8.2   Pose-only Map Joining

For 2D pose-only map joining, three 2D pose graph SLAM datasets, including one experimental dataset Intel [34] and two simulation datasets Manhattan [9] and City10000 [8], are used. For 3D pose-only map joining, two 3D pose graph SLAM datasets, including one simulation dataset Sphere [7] and one experimental dataset Parking Garage [6], are used.

(a) Feature uncertainty   (b) Local view   (c) Pose uncertainty

Fig. 10. Uncertainty estimations of both features (a and b) and poses (c) by Linear SLAM and Nonlinear LS for Victoria Park dataset. For better visualisation, the uncertainty shows in (a) and (b) is scaled by 2.

(a) Feature uncertainty   (b) Local view   (c) Pose uncertainty

Fig. 11. Uncertainty estimations of both features (a and b) and poses (c) by Linear SLAM and Nonlinear LS for DLR dataset.

Table 4
The Absolute and Relative RMSE* and χ^2 of Different Pose Graph SLAM Algorithms

| Dataset | HOG-Man | | | COP-SLAM | | | Linear SLAM | | | Nonlinear LS |
	RMSE(Abs)	RMSE(Rel)	χ^2	RMSE(Abs)	RMSE(Rel)	χ^2	RMSE(Abs)	RMSE(Rel)	χ^2	χ^2
Intel	0.054648	0.017277	972.00				**0.006571**	**0.000216**	**546.51**	546.46
Manhattan	1.180292	0.028890	548.01				**1.114862**	**0.011576**	**214.12**	137.91
City10000	**0.062695**	0.019269	1276.5				0.191676	**0.004678**	**601.38**	511.99
Sphere	1.854379	0.090491	2051.6	2.877556	0.236187	5532.6	**1.303615**	**0.050658**	**1363.8**	1154.5
ParGarage	2.315629	0.008350	2.1998	4.847823	0.273661	1268.9	**1.603590**	**0.004693**	**1.5978**	1.2888

* Both absolute RMSE (RMSE(Abs)) and relative RMSE (RMSE(Rel)) (in meters) of the pose positions are w.r.t. the results of the Nonlinear LS.

The results of 2D pose-only map joining using Linear SLAM are shown in Fig. 12(a) to Fig. 12(c) while the results of 3D pose-only map joining using Linear SLAM are shown in Fig. 12(d) to Fig. 12(e). For these five datasets, each local map is directly obtained using the relative poses with respect to one pose, thus there is no nonlinear optimization process for local map building involved in the Linear SLAM algorithm. For comparison, HOG-Man [38] and Nonlinear LS are applied to get the SLAM results for these five pose graph datasets. COP-SLAM [40] is also performed to the 3D Sphere and Park-

ing Garage datasets. The results are also shown in Fig. 12. The RMSE of both the absolute trajectory error [41] and the relative trajectory error [42], as well as the χ^2 of Linear SLAM, HOG-Man, and COP-SLAM as compared with the Nonlinear LS SLAM results are shown in Table 4. It can be seen that, almost all the RMSE and the χ^2 of the results from Linear SLAM are smaller than those from HOG-Man and COP-SLAM (only the absolute RMSE of Linear SLAM using City10000 dataset is slightly larger than that of HOG-Man).

Fig. 12. 2D and 3D pose-only map joining results of Linear SLAM (LSLAM). Fig. 12(a) to Fig. 12(c) are the results of Intel, Manhattan and City10000 2D pose graph datasets. Fig. 12(d) and Fig. 12(e) are the results of Sphere and Parking garage 3D pose graph datasets. All results are compared with those of HOG-Man and nonlinear least squares. The COP-SLAM is also compared using 3D Sphere and Parking garage datasets. The quantitive comparison is given in Table 4.

The computational cost of Linear SLAM for these five datasets are 0.22s, 0.85s, 5.01s, 2.52s and 1.50s, respectively. For the pose graph SLAM datasets, the computational cost of Linear SLAM is about 20 times cheaper than that of HOG-Man [38]. Both of the two algorithms solve the SLAM problem in a hierarchical manner. Our implementation of Linear SLAM is slower than g2o [6] mainly due to the way our code is structured (g2o code is highly optimized and is one of the fastest implementations of Nonlinear LS SLAM). In the experiments, g2o is initialized by using the results from Linear SLAM. While, Linear SLAM does not require any initial value. The COP-SLAM [40] is faster than Linear SLAM when performing the two 3D datasets, while the accuracy is the worst comparing to Linear SLAM and HOG-Man.

8.3 Feature-only Map Joining

For 2D feature-only map joining, the VicPark200 and DLR200 datasets are used. Here the local map datasets are the same as the ones used in the experiment of 2D pose-feature map joining in Section 8.1. Similar to D-SLAM map joining (DMJ) [31], a preprocessing is done to make sure there are at least 2 (for 2D) or 3 (not collinear, for 3D) common features between every two

contiguous local maps such that the local maps can be combined together (If there are not enough common features between the two local maps, then the Linear SLAM algorithm for pose-feature maps is used to join them together to form a larger local map). Then the local maps are transformed into the coordinates defined by features and all the poses are marginalized out before the map joining.

The results of 2D feature-only map joining using Linear SLAM algorithm for the two datasets are shown in Fig. 13(a) and Fig. 13(b). For comparison, the results from DMJ and Iterated DMJ (I-DMJ) [31] are also shown in the figures. Similar to I-SLSJF, the result of I-DMJ is equivalent to that of performing nonlinear least squares optimization using all the feature-only local maps as observations [31]. As shown in Fig. 13(a) and Fig. 13(b), the results of Linear SLAM are better than those of DMJ, while close to the optimal I-DMJ results.

The RMSE of Linear SLAM and DMJ as compared with the optimal I-DMJ results are shown in Table 5. The computational cost of Linear SLAM for VicPark200 and DLR200 datasets are 1.14s, 0.58s respectively, which is significantly cheaper than that of DMJ (I-DMJ) [31].

16

	(a) VicPark200	(b) DLR200	(c) 3D Simu871

Fig. 13. 2D and 3D feature-only map joining results using Linear SLAM (LSLAM). Fig. 13(a) is the result of 2D Victoria Park datasets in the coordinate frame defined by Feature 84 and 85. Fig. 13(b) is the result of 2D DLR datasets in the coordinate frame defined by Feature 367 and 369. Both the 2D results are compared with DMJ and I-DMJ [31]. Fig. 13(c) is the result of the 3D simulation with 870 local maps in the coordinate frame defined by Feature 1758, 1757 and 1359.

Table 5
RMSE* of Feature Locations for Different Feature-Only Map Joining Algorithms

	DMJ	Linear SLAM	I-DMJ
Dataset	Feature	Feature	Feature
VicPark200	0.268626	**0.193141**	0
DLR200	0.536729	**0.393460**	0
3D Simu871	N/A	0.0014850	N/A

* For joining 2D feature-only maps, the result of I-DMJ is used as the benchmark. For the 3D Simu871 dataset used in feature-only map joining, the ground truth is the benchmark.

For the 3D feature-only SLAM, the simulated 3D Simu871 dataset is used, and the result of Linear SLAM algorithm as well as the ground truth are shown in Fig. 13(c). The RMSE of Linear SLAM as compared with the ground truth is shown in Table 5. The computational cost of Linear SLAM is 2.61s.

In summary, simulation and experimental results using publicly available datasets demonstrated that the proposed Linear SLAM is consistent and efficient and can produce results very close to the full nonlinear least squares based methods. For most of the datasets, its accuracy is better than other map joining based SLAM algorithms and some efficient SLAM algorithms such as COP-SLAM and HOG-Man.

9   Discussions

The Linear SLAM algorithm proposed in this paper is based on map joining. Map joining has already been shown to be able to improve the efficiency for large-scale SLAM as well as reduce the linearization errors. Similar to recursive nonlinear estimation (e.g. [7]), map joining also solves a recursive version of the problem step by step, thus does not get stuck in local minima very of-

ten. Apart from the advantages of existing map joining based algorithms, the key advantage of Linear SLAM is that it only requires solving linear least squares instead of nonlinear least squares. Thus there is no need to compute initial value and no need to perform iterations. The key to make the map joining problem linear is that the two maps to be fused are always built in the same coordinate frame. Basically, when joining two local maps built at different coordinate frames, the estimation/optimization problem is nonlinear, which is the case in most of the existing map joining algorithms. But when joining two local maps with the same coordinate frame, the problem can be linear, which is the idea used in this paper. The sequential map joining [13–15] and Divide and Conquer map joining [12]) join two maps at a time, thus both strategies can be applied in Linear SLAM.

Similar to SLSJF [15], CF-SLAM [35] and other optimization based submap joining approaches [31, 38], the sparseness of the information matrix is maintained in the proposed linear map joining algorithm as the coefficient matrix A in (15) is always sparse. Also similar to SLSJF and CF-SLAM, there is no information loss or information reuse in the Linear SLAM algorithm. The simulation and experimental results demonstrated that Linear SLAM results are better than that of CF-SLAM and SLSJF in most of the cases. The reason is, when joining two maps, CF-SLAM and SLSJF use EIF while Linear SLAM performs the optimal linear least squares. The Linear SLAM result is not as good as that of I-SLSJF because an additional smoothing step through nonlinear optimization is applied in I-SLSJF. It is clear that fusing local maps built in different coordinate frames optimally using linear approach is impossible. Thus if we want to apply smoothing after joining all the local maps together by Linear SLAM, we have to use nonlinear optimization.

The difference between the result of Linear SLAM and

the best solution to the full nonlinear least squares SLAM comes from two reasons: (i) Instead of using the original odometry and observation information, we summarize the local map information as the local map estimate together with its uncertainty (information matrix) and use this information in the map joining. This is the case for many map joining algorithms including CF-SLAM, SLSJF and I-SLSJF. (ii) Instead of fusing all the local maps together in one go using nonlinear optimization (as in I-SLSJF), we fuse two maps at a time (as in CF-SLAM and SLSJF), resulting in a suboptimal solution. It should be pointed out that the difference is due to the linearization performed (Linear SLAM, CF-SLAM, SLSJF, I-SLSJF are all equivalent to full least squares SLAM when the odometry and observation functions are linear).

In general, the more accurate the local maps are, the closer the Linear SLAM result to the best solution to the full nonlinear least squares SLAM. In some of the experimental results shown in Section 8 (e.g. all the Linear SLAM results using the pose graph SLAM datasets), one local map was directly obtained using the information related to one pose. In this case, Linear SLAM completely avoids the nonlinear optimization in the local map building process. However, its performance may be sacrificed because the quality of the local maps is not well controlled. To improve the performance of Linear SLAM, we recommend to use nonlinear optimization techniques to build high quality small local maps (accurate estimate and small covariance matrix) taking into account the connectivity of the local map graph [43], and then apply our linear map joining algorithm. Furthermore, the solution from Linear SLAM (using least squares to build local maps without marginalization) can be used as an (excellent) initial guess for the iterative approaches to solve the full nonlinear least squares SLAM, similar to T-SAM2 [44] or [39]. However, this will add more nonlinear components in the algorithm. The tradeoff between improving the accuracy of the result and reducing the nonlinear components in Linear SLAM algorithm requires further investigations.

It should also be pointed out that the three local maps in Fig. 1(b), Fig. 1(c) and Fig. 1(d) are all optimal for the given local map data; they are equivalent and can be transferred from one to another using coordinate transformations. However, when using these different versions of local maps in map joining, the final results can be slightly different due to the different linearization performed. For the actual nonlinear odometry and observation models, using which version of local maps gives the best solution depends on many factors (such as the size and number of local maps, the number of features observed at each step, the noise level of odometry and observation, etc.) and requires further investigations.

We would like to point to literature [20, 21] indicating that SLAM problem is close to linear and its dimensionality can be substantially reduced as possible reasons why the linear approximation proposed in this paper produces solutions that are very close to those obtained using optimal nonlinear least squares SLAM. Linear SLAM is a nice way to separate the linear components from the nonlinear components in the SLAM problem through submap joining.

Unlike traditional local submap joining, the final global map obtained from this Linear SLAM algorithm is not in the coordinate frame of the first local map. For Divide and Conquer map joining, the coordinate frame of the final global map is the same as the coordinate frame of the last two maps that were fused together. For sequential map joining, the coordinate frame of the final global map is the last robot pose in the last local map. This is somewhat similar to the robocentric mapping [45] where the map is transformed into the coordinate frame of the current robot pose in order to reduce the linearization error in the EKF framework. As discussed in Section 3.1, if a final global map in the coordinate frame of the first robot pose is required, e.g. in order to compare the result with traditional local submap joining, only a coordinate transformation described in Section 4.3 is needed. The global map after the coordinate transformation is equivalent to the global map obtained from the Linear SLAM algorithm.

Using the proposed Linear SLAM framework, the pose graph SLAM and the D-SLAM mapping can be solved in the same way as feature-based SLAM. The differences are the treatments of orientation of poses and the coordinate transformations. The feature-only map joining requires the coordinate transformation to be done where the coordinate frames are defined by 2 (for 2D cases) or 3 features (for 3D cases). In pose graph SLAM, there can be a number of common poses in the two pose graphs to be fused. Thus the wraparound of robot orientation angles need to be considered. Because we only fuse two maps at a time, the wraparound issue can be dealt with very easily. This is different from the wraparound issue in the linear approximation approach proposed in [29, 46]. Some other limitations of the linear method in [29] are: (1) it can only be applied to 2D pose graph SLAM; (2) it requires special structure on the covariance matrices.

Data association is not considered in the experimental results presented in this paper. However, since our Linear SLAM approach is using linear least squares and the associated information matrices are always available, many of the data association methods suggested for EIF based SLAM algorithms (such as SLSJF [15], CF-SLAM [35] and ESEIF [47]), or optimization based SLAM algorithms (such as iSAM [7]), including the strategies for covariance submatrix recovery, can be applied together with the proposed Linear SLAM algorithm.

10   Conclusion

This paper demonstrated that submap joining can be implemented in a way such that only solving linear least squares problems and performing coordinate transformations are needed. There is no assumption on the structure of local map covariance matrices and the approach can be applied to feature-based SLAM, pose graph SLAM, as well as D-SLAM, for both 2D and 3D scenarios. Since closed-form solutions exist for linear least squares problems, this new approach avoids the requirements of accurate initial value and iterations which are necessary in most of the existing nonlinear optimization based SLAM algorithms.

Simulation and experimental results using publicly available datasets demonstrated the consistency, efficiency and accuracy of the Linear SLAM algorithms, and that Linear SLAM outperforms some other map joining based SLAM algorithms and some recent developed efficient SLAM algorithms. Although the result of Linear SLAM looks very promising, it is not equivalent to the optimal solution to SLAM based on nonlinear least squares starting from an accurate initial value. Like some other submap joining algorithms, how different the Linear SLAM result is from the optimal solution depends on many factors such as the quality of the original sensor data as well as the size and quality of the local maps. Further research is necessary to analyze this in details and develop smart strategies for separating SLAM data for building local maps in order to optimize the performance of Linear SLAM.

References

[1] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229-241, 2001.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J.J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309-1332, 2016.

[3] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333-349, 1997.

[4] 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, pp. 1181-1203, 2006.

[5] K. Konolige, G. Grisetti, R. Kummerle, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, pp. 22-29, 2010.

[6] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3607-3613, 2011.

[7] M. Kaess, A. Ranganathan, and F. Dellaert. "iSAM: Incremental smoothing and mapping". *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365-1378, 2008.

[8] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, pp. 217-236, 2012.

[9] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2262-2269, 2006.

[10] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428-439, 2009.

[11] S. Huang, Y. Lai, U. Frese, and G. Dissanayake, "How far is SLAM from a linear least squares problem?" In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, pp. 3011-3016, 2010.

[12] L. M. Paz, J. D. Tardos, and J. Neira, "Divide and Conquer: EKF SLAM in O(n)," *IEEE Transactions on Robotics,* vol. 24, no. 5, pp. 1107-1120, 2008.

[13] 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, pp. 311-330, 2002.

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

[15] S. Huang, Z. Wang, and G. Dissanayake, "Sparse local submap joining filter for building large-scale maps," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1121-1130, 2008.

[16] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller, "SLAM in large-scale cyclic environments using the atlas framework," *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113-1139, 2004.

[17] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3185-3192, 2010.

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

[19] S. Huang, Z. Wang, G. Dissanayake, and U. Frese, "Iterated SLSJF: A sparse local submap joining algorithm with improved consistency," In *Proc. Australasian Conference on Robotics and Automation*, 2008.

[20] S. Huang, H. Wang, U. Frese, and G. Dissanayake, "On the number of local minima to the point feature based SLAM problem," In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2074-2079, 2012.

[21] H. Wang, G. Hu, S. Huang, and G. Dissanayake, "On the structure of nonlinearities in pose graph SLAM," In *Proc. Robotics: Science and Systems (RSS)*, 2012.

[22] D. M. Rosen, L. Carlone, A. S. Bandeira and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," In *Proc. 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[23] Y. S. Hwang and J. M. Lee, "Robust 2D map building with motion-free ICP algorithm for mobile robot navigation," *Robotica*, vol. 35, no. 9, pp. 1845-1863, 2017.

[24] A. V. Savkin and H. Li, "A safe area search and map building algorithm for a wheeled mobile robot in complex unknown cluttered environments," *Robotica*, vol. 36, no. 1, pp. 96-118, 2018.

[25] H. Wang, S. Huang, K. Khosoussi, U. Frese, G. Dissanayake and B. Liu, Dimension reduction for point feature SLAM problems with spherical covariance matrices, *Automatica*, vol. 51, pp. 149-157, 2015.

[26] L. Zhao, S. Huang, and G. Dissanayake, "Linear SLAM: A linear solution to the feature based and pose graph SLAM based on submap joining" In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, pp. 24-30, 2013.

[27] L. Carlone, and F. Dellaert. Duality-based verification techniques for 2D SLAM. In Proc. International Conference on Robotics and Automation (ICRA), pp. 4589-4596, 2015.

[28] Z. Wang, S. Huang, and G. Dissanayake, "D-SLAM: A decoupled solution to simultaneous localization and mapping". *International Journal of Robotics Research*, 26(2), 187-204, 2007.

[29] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A linear approximation for graph-based simultaneous localization and mapping," In *Proc. Robotics: Science and Systems (RSS)*, 2011.

[30] L. Carlone and A. Censi, "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 475-492, 2014.

[31] S. Huang, Z. Wang, G. Dissanayake, and U. Frese, "Iterated D-SLAM map joining: Evaluating its performance in terms of consistency, accuracy and efficiency". *Autonomous Robots*, 27(4), 409-429, 2009.

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

[33] J. Kurlbaum and U. Frese, "A benchmark data set for data association," Technical Report, University of Bremen, available online: http://www.sfbtr8.uni-bremen.de/reports.htm, Data available online http://radish.sourceforge.net/.

[34] D. Hahnel, W. Burgard, D. Fox and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 206-211, 2003.

[35] C. Cadena and J. Neira, "SLAM in O(log n) with the combined Kalman-information filter," *Robotics and Autonomous Systems*, vol. 58, no. 11, pp. 1207-1219, 2010.

[36] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3562-3568, 2006.

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

[38] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 273-278, 2010.

[39] G. Grisetti, R. Kummerle, and K. Ni, "Robust optimization of factor graphs by using condensed measurements," In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 581-588, 2012.

[40] G. Dubbelman, and B. Brownig, "COP-SLAM: closed-form online pose-chain optimization for visual SLAM," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1194-1213, 2015.

[41] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system". In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1691-1696, 2012.

[42] R. Kummerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. "On measuring the accuracy of SLAM algorithms". *Autonomous Robots*, 27(4), 387-407, 2009.

[43] E. Olson and M. Kaess, M. "Evaluating the performance of map optimization algorithms". *In RSS Workshop on Good Experimental Methodology in Robotics (Vol. 15)*. June 2009.

[44] K. Ni and F. Dellaert, "Multi-level submap based SLAM using nested dissection," In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2558-2565, 2010.

[45] 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*, vol. 55, no. 1, pp. 21-29, 2007.

[46] J. L. Martinez, J. Morales, A. Mandow, and A. Garcia-Cerezo, "Incremental closed-form solution to globally consistent 2D range scan mapping with two-step pose estimation," In *Proc. The 11th IEEE International Workshop on Advanced Motion Control*, pp. 252-257, 2010.

[47] 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, pp. 335-359, 2007.

[48] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser and P. Sayd, "Generic and Real Time Structure from Motion using Local Bundle Adjustment," *Image and Vision Computing*, vol. 27, no. 8, pp. 1178-1193, 2009.

A Proof of Lemma 1

In this appendix, we provide a proof of Lemma 1 in Section 4.3. Here we use some notations in Section 4.

Suppose $(\mathbf{Z}, I_Z)$ are the information from the two local maps defined in (13) and (14). The unique optimal solution to minimize (12) is given by (16) and the corresponding information matrix is given by (17).

That is, for any $\bar{\mathbf{X}}^{G_{12}} \neq \hat{\bar{\mathbf{X}}}^{G_{12}}$,

$$\|Z - A\bar{\mathbf{X}}^{G_{12}}\|_{I_Z}^2 > \|Z - A\hat{\bar{\mathbf{X}}}^{G_{12}}\|_{I_Z}^2. \qquad (A.1)$$

The one-to-one transformation function that transforming between the state vector $\bar{\mathbf{X}}^{G_{12}}$ and $\mathbf{X}^{G_{12}}$ are given in (9) and (18).

The nonlinear least squares problem for building the global map $M^{G_{12}}$ in the coordinate frame of the end pose of local map 2 is to minimize (7), which can be reformulated as

$$\|Z - Ag(\mathbf{X}^{G_{12}})\|_{I_Z}^2. \qquad (A.2)$$

Fig. A.1. Definition of coordinate frame in 2D feature-only map.

Fig. B.1. Definition of coordinate frame in 3D feature-only map.

The unique optimal solution to minimize (A.2) must be

$$\hat{\mathbf{X}}^{G_{12}} = g^{-1}(\hat{\bar{\mathbf{X}}}^{G_{12}}). \qquad (A.3)$$

Because for any $\mathbf{X}^{G_{12}} \neq g^{-1}(\hat{\bar{\mathbf{X}}}^{G_{12}})$, $g(\mathbf{X}^{G_{12}}) \neq \hat{\bar{\mathbf{X}}}^{G_{12}}$, and thus

$$\|Z - Ag(\mathbf{X}^{G_{12}})\|_{I_Z}^2 \neq \|Z - A\hat{\bar{\mathbf{X}}}^{G_{12}}\|_{I_Z}^2. \qquad (A.4)$$

Since $\|Z - A\hat{\bar{\mathbf{X}}}^{G_{12}}\|_{I_Z}^2$ is the minimal objective function value, thus

$$\|Z - Ag(\mathbf{X}^{G_{12}})\|_{I_Z}^2 > \|Z - A\hat{\bar{\mathbf{X}}}^{G_{12}}\|_{I_Z}^2 = \|Z - Ag(\hat{\mathbf{X}}^{G_{12}})\|_{I_Z}^2. \qquad (A.5)$$

Now we have proved that the optimal solution to (A.2) must be (A.3). That is, applying the coordinate transformation on the linear least squares solution.

Because the Jacobian J of $Ag(\mathbf{X}^{G_{12}})$ with respect to $\mathbf{X}^{G_{12}}$, evaluated at $\hat{\mathbf{X}}^{G_{12}}$ can be computed by

$$J = A\nabla \qquad (A.6)$$

where ∇ is the Jacobian of $g(\mathbf{X}^{G_{12}})$ with respect to $\mathbf{X}^{G_{12}}$, evaluated at $\hat{\mathbf{X}}^{G_{12}}$, as given in (21), thus the corresponding information matrix can be computed as

$$I^{G_{12}} = J^T I_Z J = \nabla^T A^T I_Z A \nabla = \nabla^T \bar{I}^{G_{12}} \nabla \qquad (A.7)$$

which is equivalent to computing the information matrix through the information matrix of $\hat{\bar{\mathbf{X}}}^{G_{12}}$ as in (20).

Thus we have proved Lemma 1.

B   Formula of Coordinate Transformation for Feature-only Maps

In this appendix, we provide the details about the definition of the coordinate frame and the coordinates transformation for feature-only map, in 2D and 3D scenarios.

B.1   Coordinate Frame in 2D Feature-only Map

In a 2D feature-only map, two features are used to define the coordinate frame of the map.

Suppose the two features which are used to define the coordinate frame are $\mathbf{F}_O$ and $\mathbf{F}_X$. We define $\mathbf{F}_O$ as the origin of the map, and define the direction from feature $\mathbf{F}_O$ to $\mathbf{F}_X$ (the vector $\mathbf{F}_X - \mathbf{F}_O$) as the x axis of the coordinate frame. The coordinate frame defined by feature $\mathbf{F}_O$ and $\mathbf{F}_X$ is shown in Fig. A.1.

By definition of the coordinate frame using $\mathbf{F}_O$ and $\mathbf{F}_X$, $\mathbf{F}_O = [0,0]^T$ and the y coordinate of $\mathbf{F}_X$ is 0. These three parameters are not included in the state vector of the feature-only map.

B.2   Coordinate Frame in 3D Feature-only Map

In a 3D feature-only map, three features are used to define the coordinate frame of the map.

Suppose the three features which are used to define the coordinate frame are $\mathbf{F}_O$, $\mathbf{F}_X$ and $\mathbf{F}_Y$. We first define $\mathbf{F}_O$ as the origin of the map and define the direction from feature $\mathbf{F}_O$ to $\mathbf{F}_X$ (the vector $\mathbf{F}_X - \mathbf{F}_O$) as the x axis of the coordinate frame. Then the x-y plane is defined by the x axis and the vector $\mathbf{F}_Y - \mathbf{F}_O$. Thus the z axis can be defined by the cross product of the x axis and $\mathbf{F}_Y - \mathbf{F}_O$, which is perpendicular to the x-y plane. Then, the y axis can be defined by the cross product of z and x axis. The coordinate frame defined by feature $\mathbf{F}_O$, $\mathbf{F}_X$ and $\mathbf{F}_Y$ is shown in Fig. B.1.

Because $\mathbf{F}_X - \mathbf{F}_O$ and $\mathbf{F}_Y - \mathbf{F}_O$ are used to define the x-y plane, the three features which define the coordinate frame of the 3D feature-only map cannot be collinear.

By definition of the coordinate frame using $\mathbf{F}_O$, $\mathbf{F}_X$ and $\mathbf{F}_Y$, $\mathbf{F}_O$ is the origin thus $\mathbf{F}_O = [0,0,0]^T$. $\mathbf{F}_X$ is on the x axis thus the y and z coordinates of $\mathbf{F}_X$ are 0. $\mathbf{F}_Y$ is in the $x - y$ plane thus the z coordinate of $\mathbf{F}_Y$ is 0. These six parameters defines the 6 DOF 3D coordinate frame, thus are not included in the state vector of the feature-only map.

B.3 Coordinate Transformation for 2D Feature-only Map

Suppose in the current coordinate frame, $\mathbf{F}_O = [x_{F_O}, y_{F_O}]^T$, $\mathbf{F}_X = [x_{F_X}, y_{F_X}]^T$ and a feature position is $\mathbf{F}$. In the new coordinate frame defined by $\mathbf{F}_O$ and $\mathbf{F}_X$ as described in Appendix B.1, the new coordinates of the same feature, $\mathbf{F}'$, can be obtained by a coordinate transformation

$$\mathbf{F}' = R\,(\mathbf{F} - \mathbf{t}). \tag{B.1}$$

From Fig. A.1, it is easy to see that the translation $\mathbf{t} = \mathbf{F}_O$. And the rotation angle ϕ from the current coordinate frame to the new coordinate frame can be computed as

$$\phi = \text{atan2}\,(y_{F_X} - y_{F_O},\ x_{F_X} - x_{F_O}). \tag{B.2}$$

Thus the rotation matrix $R = r(\phi)$ where $r(\cdot)$ is the angle-to-matrix function.

B.4 Coordinate Transformation for 3D Feature-only Map

Now we consider the 3D coordinate transformation from the current coordinate frame to the coordinate frame defined by the features $\mathbf{F}_O$, $\mathbf{F}_X$ and $\mathbf{F}_Y$.

The coordinate transformation in 3D is also in the format of (B.1). The translation in the 3D coordinate transformation is also $\mathbf{t} = \mathbf{F}_O$.

For the rotations, suppose vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are computed as

$$\begin{cases} \mathbf{v}_1 = \mathbf{F}_X - \mathbf{F}_O \\ \mathbf{v}_3 = \mathbf{v}_1 \times (\mathbf{F}_Y - \mathbf{F}_O) \\ \mathbf{v}_2 = \mathbf{v}_3 \times \mathbf{v}_1 \end{cases} \tag{B.3}$$

As described in Appendix B.2, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are the vectors in the direction of the x, y and z axis of the coordinate frame defined by the features $\mathbf{F}_O$, $\mathbf{F}_X$ and

$\mathbf{F}_Y$, respectively. Thus we have

$$\begin{cases} R\,\mathbf{v}_1 = [\|\mathbf{v}_1\|, 0, 0]^T \\ R\,\mathbf{v}_2 = [0, \|\mathbf{v}_2\|, 0]^T \\ R\,\mathbf{v}_3 = [0, 0, \|\mathbf{v}_3\|]^T. \end{cases} \tag{B.4}$$

So the rotation matrix R in (B.1) is given by

$$R = [\mathbf{v}_1/\|\mathbf{v}_1\|, \mathbf{v}_2/\|\mathbf{v}_2\|, \mathbf{v}_3/\|\mathbf{v}_3\|]^T. \tag{B.5}$$