Exactly Sparse Delayed State Filter on Lie groups for Long-term Pose Graph SLAM

The International Journal of Robotics Research
XX(X):1–27

DOI: 10.1177/ToBeAssigned
www.sagepub.com/

Kruno Lenac, Josip Ćesić, Ivan Marković and Ivan Petrović[1]

Abstract

In this paper we propose a SLAM back-end solution called the exactly sparse delayed state filter on Lie groups (LG-ESDSF). We derive LG-ESDSF and demonstrate that it retains all the good characteristics of the classic Euclidean ESDSF—main being the exact sparsity of the information matrix. The key advantage of LG-ESDSF in comparison to the classic ESDSF lies in the ability to respect the state space geometry by negotiating uncertainties and employing filtering equations directly on Lie groups. We also exploit the special structure of the information matrix in order to allow long-term operation while the robot is moving repeatedly through the same environment. To prove the effectiveness of the proposed SLAM solution, we conducted extensive experiments on two different publicly available datasets, namely the KITTI and EuRoC dataset, using two front-ends: one based on the stereo camera and the other on the 3D LIDAR. We compare LG-ESDSF with the general graph optimization framework (g^{2}o) when coupled with the same front-ends. Similarly to g^{2}o the proposed LG-ESDSF is front-end agnostic and the comparison demonstrates that our solution can match the accuracy of g^{2}o, while maintaining faster computation times. Furthermore, the proposed back-end coupled with the stereo camera front-end forms a complete visual SLAM solution dubbed LG-SLAM. In the end, we evaluated LG-SLAM using the online KITTI protocol and at the time of writing it achieved the second best result among the stereo odometry solutions and the best result among the tested SLAM algorithms.

Keywords

SLAM, exactly sparse delayed state filter, Lie groups, graph optimization

1 Introduction

Mobile robots are about to become omnipresent. Starting from our homes, where they could help us in everyday chores, to search and rescue operations where they could replace us in dangerous tasks, and finishing with factories where, even nowadays, they increase safety, production speed, and performance. Majority of these robots will be capable to autonomously complete complex tasks without human intervention in changing and unknown environments. Regardless of the given task, every autonomous mobile robot must have at its disposal (i) a map of the environment, and (ii) its location within the map. Since fully autonomous mobile robots should not rely on a pre-built environment map, they should be able to simultaneously build the map of the environment and localize within the same map. This problem is known in mobile robotics as simultaneous localization and mapping (SLAM) and was one of the most intensively researched problems during the past two decades. For more information on the fundamentals of the SLAM the reader is referred to Durrant-Whyte and Bailey (2006); Bailey and Durrant-Whyte (2006), while a more detailed analysis of SLAM observability and convergence can be found in Dissanayake et al. (2011).

From the first solutions to the SLAM problem until today, a large number of different SLAM algorithms has been presented Cadena et al. (2016). In general, a SLAM algorithm can be divided in two main parts: (i) the SLAM *front-end* and (ii) the *SLAM back-end*. The SLAM front-end is responsible for dealing with sensor data abstractions, e.g., extracting features and constraints, while the SLAM back-end is responsible for estimation and optimization of both

[1]University of Zagreb Faculty of Electrical Engineering and Computing, Croatia

Corresponding author:
Kruno Lenac, University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia.
Web: www.fer.unizg.hr/en/kruno.lenac
Email: kruno.lenac@fer.hr

robot and map landmarks poses based on the SLAM front-end constraints. In most SLAM algorithms, the SLAM back-end is independent of the SLAM front-end, thus allowing for the same front-end to be used with different SLAM back-ends and vice-versa, depending on the system requirements and available sensors. In the sequel we focus our discussion to SLAM back-ends, more specifically on the pose graph approaches, since the major part of this paper and its contributions focus on this subject.

SLAM back-ends can be divided in two groups depending on the approach to state optimization: (i) filtering based (e.g. Aulinas et al. (2008)) and (ii) graph-optimization based SLAM back-ends (e.g. Grisetti et al. (2010)). Recently, works in the field of optimization based SLAM have been specifically focusing on various optimization related components, including initialization, convergence and global optimality aspects. The initialization aspects emphasize the problem arising when non-Euclidean variables exist in the system, such as rotation Carlone et al. (2015) or full pose estimation Arrigoni et al. (2016). Also, once the initialization is performed, convergence Carlone (2013) and global optimality Briales and Gonzalez-Jimenez (2016, 2017) aspects of the problem can be analyzed. Filtering based SLAM back-ends can be divided based on the states used for the map and location estimation. The first group are the feature-based SLAM back-ends that estimate the current robot pose and poses of extracted map landmarks. Given that, map landmarks and the robot location are correlated and must be updated in the same step. The second group of filtering based approaches are called pose graph SLAM back-ends. These approaches estimate a discrete robot trajectory, while map landmarks are correlated only to one of the discrete trajectory states. The result is that map landmarks are no longer correlated to each other, but only to a single trajectory state which then allows trajectory estimation being independent of the environment map estimation.

The approach that we propose in the present paper falls under the category of filtering based pose graph SLAM back-ends. Herein, we draw upon our earlier work presented in a conference paper (Lenac et al. (2017)), where we proposed a preliminary version of the novel filter on Lie groups and experimental results comparing our SLAM solution to state-of-the-art visual SLAM algorithms. The main contribution of the present paper is the introduction of a long-term pose graph SLAM implemented on Lie groups. The Lie group implementation allows our SLAM system to respect the state space geometry and thus achieve state-of-the-art performance comparable to that of a popular graph-optimization approach, namely g^2o, but with smaller computation time demonstrated on extensive experiments using two different publicly available datasets, namely the KITTI and EuRoC dataset. Our approach also retains all the good characteristics of the pose graph SLAM from which the most important two are sparsity of the information matrix and the separation between trajectory and map estimation.

The rest of the paper is organized as follows. In Section 2 we give a survey of the related research and elaborate the contributions of the proposed solution in details. In Section 3 we give an overview of the standard Euclidean filter our novel SLAM solution is based on. Section 4 gives introduction to the Lie group fundamentals required for deriving the novel filter, while the SLAM based on the novel filter is presented in Section 5. Section 6 presents modifications applied to our SLAM system in order to allow long-term autonomy. In the end, experimental results are given in Section 7, while Section 8 concludes the paper.

2 Related research

Although the graph optimization approach to SLAM was known and well defined in the early stages of SLAM development, first SLAM solutions were nevertheless based on the filtering approach. The main reason behind this, at the time, was the inability to compute graph optimization in the time required for successful SLAM operation. The first filtering based SLAM solutions utilized the extended Kalman filter (EKF); however, issues were detected stemming from the linearization of both process and measurement model and from the increasing number of states when new landmarks are extracted and added as new filter states. Through time, many efficient implementations of the EKF-SLAM were presented, such as Dissanayake et al. (2001); Guivant and Nebot (2001, 2002); Tardós et al. (2002); Kohlhepp et al. (2004); Weingarten and Siegwart (2006); Davison et al. (2007); Civera et al. (2010), but the core problems remained. The first fundamentally different approach to EKF-SLAM was presented in the form of a particle filter (PF) SLAM, dubbed Fast-SLAM in Montemerlo et al. (2002). The main advantage of the PF is that there is no need to linearize the system model, while the main drawback is high dimensionality of the state space. This was solved in Fast-SLAM by applying Rao-Blackwell marginalization. The improved version of the Fast-SLAM, dubbed Fast-SLAM 2.0, was later presented in Montemerlo et al. (2003).

In order to solve the issue of high dimensionality of the EKF-SLAM state-space, researches turned to the information form of the EKF called the extended information filter (EIF). The main advantage of using EIF in SLAM is that with a large numbers of landmarks in the state-space, most of the off-diagonal elements of the information matrix are close to zero. One of the first successful solutions for the EIF-SLAM was presented in Thrun et al. (2004) and more were presented thereafter

in Wang et al. (2006); Walter et al. (2007). However, disregarding elements that are close to zero leads to inevitable introduction of estimation errors, which in most cases for a well designed system are small, but still do exist. In Eustice et al. (2006) authors presented a solution to the information-form SLAM that has an exactly sparse information matrix and thus no approximation error occurs. Therein, they named the developed information filter the exactly sparse delayed state filter (ESDSF).

The sparsity of the SLAM matrix was also a key insight that allowed developing new direct linear solvers for the SLAM problem using graph optimization techniques, such as in Davis (2006). New solvers allowed the refinement process to complete tens of times faster than before, which opened a whole new research area for SLAM algorithms. One of the first successful alternatives to filtering approaches, dubbed $\sqrt{\text{SLAM}}$, was presented in Dellaert and Kaess (2006). $\sqrt{\text{SLAM}}$ used a smoothing approach to solve SLAM and achieved better performance in both computation time and accuracy than the contemporary existing EKF-SLAM solutions. In Konolige et al. (2010) authors presented a method for optimizing large pose graphs called the sparse pose adjustment (SPA). SPA is similar to $\sqrt{\text{SLAM}}$, but with the main differences in (i) the efficient construction of the linear subproblem, by employing ordered data structures, and (ii) in using the Levenberg–Marquardt (LM) algorithm instead of the nonlinear least squares. A graph optimization solution was presented in Konolige (2010) which used an efficient version of the sparse bundle adjustment (SBA). Therein, relations among cameras are also sparse and, by combining the proposed method with direct sparse Cholesky solvers, authors outperformed the standard SBA implementations. A stereo SLAM solution, named S-PTAM, was presented in Pire et al. (2015) and used a parallel estimation process of the map and robot poses, thus enabling fast computation of robot location, while building map and refining the graph in the background. S-PTAM uses the LM optimization for refinement, while binary features are used for describing visual point landmarks. A graph based SLAM solution was also presented in Bourmaud and Mégret (2015), where to speed up computation and allow execution in large scale environments, authors divided the global graph into subgraphs optimized independently using the LM algorithm. Subgraphs are then matched and combined into global graph using loopy belief propagation algorithm called large scale relative similarity averaging.

In parallel to various complete SLAM solutions that used graph optimization and comprised of both the front-end and back-end, researchers started to develop universal graph optimization solutions. These solutions can be used as SLAM back-ends for trajectory and map optimization, as well as for any optimization problem that can be formulated using a graph structure. Currently, there exists several such solutions including GTSAM of Dellaert (2012), Ceres of Agarwal et al. (2010) and SLAM++ of Polok et al. (2013), but the two solutions most commonly used in combination with different SLAM front-ends are iSAM of Kaess et al. (2008) and g^2o of Kuemmerle et al. (2011). iSAM uses a fast and incremental QR matrix factorization. By updating only the QR factorization of the sparse smoothing information matrix, it recalculates only those matrix elements that change drastically, thus increasing the computation speed. Another positive aspect of such matrix factorization is an easy and fast access to estimation uncertainties. Upgrade of the iSAM, called iSAM2, was presented in Kaess et al. (2012) which introduced a new data structure called the Bayes tree.

Among all the aforementioned solutions, g^2o is the most general optimization framework for nonlinear least squares problems that can be represented as a graph. It was developed to utilize sparse connectivity in the graph, and also to take the advantage of the special structures that occur in the graph when being built by specific algorithms like the graph SLAM or bundle adjustment (BA) SLAM. As a graph optimization solution, g^2o is highly efficient and computationally fast thanks to using advanced methods for solving sparse systems and advanced features of modern processors as well as maximally optimizing processor memory and cache usage. Similarly to the other solutions, g^2o is also publicly available, but currently enjoys better community support and offers a large database of tutorials and examples. Therefore, it is not surprising that the two current state-of-the-art visual SLAM solutions, namely ORB-SLAM and LSD-SLAM, both use g^2o as their SLAM back-end. LSD-SLAM was first presented for mono-cameras in Engel et al. (2014) and afterwards a stereo-solution in Engel et al. (2015) was introduced. It employs a direct and featureless method which minimizes photometric errors between images in order to estimate the pose. Main novelties included direct tracking method which operated on $\mathsf{Sim}(3)$ and probabilistic solution to include the effect of noisy depth values into tracking. ORB-SLAM was also first introduced for mono cameras in Mur-Artal et al. (2015), and later for stereo and RGB-D cameras in Mur-Artal and Tardós (2017). It uses ORB features for mapping, loop closing and tracking, and employs covisibility graph and survival of the fittest strategy to allow for real-time execution over long periods of time in large-scale environments.

By looking at the past decade, one can notice the dominant popularity of graph optimization back-end solutions over the filtering approaches. However, regardless of the SLAM version, we always have the need to estimate the robot pose and poses of map landmarks which inherently in 3D reside on $\mathsf{SE}(3)$. Filtering solutions

dominantly relied on the Euler angles or quaternions for representing these poses. Although sufficient, filtering with these representations within Euclidean frameworks does not represent the natural way of characterizing uncertainties and relations between the state vector elements. For example, a more natural way to characterize uncertainties over unit quaternions is the Bingham distribution on the unit hypersphere (see works of Gilitschenski et al. (2015) and Glover and Kaelbling (2013)). This is in contrast to the state-of-the-art graph optimization back-ends, which relied more on using the insights of Lie groups and Lie algebras within the framework. We believe this to be one of the main reasons why filtering approaches were generally not on par with the graph-based optimization based SLAM performance. This is not to say that filtering approaches could not achieve state-of-the-art performance. The work of Mourikis and Roumeliotis (2007) investigated the problem of vision-aided inertial navigation, wherein the authors used an EKF-based approach by deriving a measurement model that avoids including the 3D feature position in the state vector, but is able to express the geometric constraints. The attitude of the sensor was modeled using unit quaternions, but the approach relied on EKF filtering of the error state using a window of a limited number of poses, hence did not aim to solve a SLAM problem including loop closings. Furthermore, some filtering approaches did utilize insights from Lie groups and applied it, to an extent, on the EKF and PF SLAM, as in Eade (2008), but in the end, resorted to the graph optimization framework and used Lie group insights to define and manipulated the graph edges, as in Eade et al. (2010). In the end, although some modifications were done to the filter itself, what was missing was a deeper structural change to the EKF itself. Changes in this field began to happen only recently with the introduction of the EKF on Lie groups (LG-EKF) of Bourmaud et al. (2013). By representing the states on Lie groups, and performing filtering equations in the pertaining Lie algebra, LG-EKF is able to respect the geometry of the state space, thus achieving greater estimation accuracy of both the mean and the covariance. Solutions to the unscented Kalman filter on Lie groups (LG-UKF) also appeared in Hertzberg et al. (2013), then were followed by the continuous-discrete EKF on Lie groups in Bourmaud et al. (2015), and invariant filters on Lie groups of Barrau and Bonnabel (2015). However, in order to bear the potential for state-of-the-art SLAM performance, the solution to the information form of the LG-EKF, i.e., the extended information filter on Lie groups (LG-EIF), was still missing; nevertheless, this was solved recently in our previous work presented in Ćesić et al. (2017). Now, with the LG-EIF as the basis, we are able to develop a SLAM solution that is not only capable of using sparse structure of the SLAM information form, but which also respects the state space geometry by representing states on Lie groups.

In this paper we draw upon our earlier work presented in a conference paper (Lenac et al. (2017)) and propose and derive equations for the novel ESDSF on Lie groups (LG-ESDSF). LG-ESDSF retains all the good characteristics of the classic ESDSF implementation, but also respects the state space geometry by negotiating uncertainties and employing filtering equations on Lie groups. Furthermore, we propose a SLAM solution based on the LG-ESDSF back-end, which in the rest of this paper we refer to as LG-SLAM, and demonstrate that it is able to attain state-of-the-art performance comparable to a graph optimization approach, namely g^{2}o. For this purpose we have coupled both g^{2}o and LG-SLAM with two different front-ends, one based on the stereo camera (Cvišić and Petrović (2015)) and one based on the 3D LIDAR (Stoyanov et al. (2012)), and tested them on two large and quite different publicly available datasets: the KITTI vision benchmark suite Geiger et al. (2012) and EuRoC dataset Burri et al. (2016). We also compare LG-SLAM based on the stereo vision front-end of Cvišić and Petrović (2015) to ORB-SLAM and LSD-SLAM on the KITTI and EuRoC datasets, and have submitted our solution for online evaluation. At the time of writing, LG-SLAM ranks second among the stereo vision approaches and first among all the tested SLAM solutions. Moreover, we have developed long-term capability for LG-SLAM by allowing it to maintain low number of trajectory states while repeatedly moving through the same environment. This solution is solely based on the information available to the SLAM back-end and does not in any way rely on the front-end. However, this is not a strict limitation—the solution can be modified to achieve even better performance once the particular front-end has been selected.

3 Euclidean ESDSF SLAM

ESDSF was first proposed in Eustice et al. (2006). It is a special form of EIF with the main advantage of having an exactly sparse information matrix. The SLAM back-end based on ESDSF is a pose graph SLAM which, as explained above, allows trajectory estimation independent of the environment map. Since the map estimation is not directly dependent on the SLAM back-end, in the present paper we do not discuss the mapping components and focus only on the trajectory estimation. Trajectory T_n in an ESDSF SLAM consists of n discrete states X_i, $i = 1 \dots n$, and is represented by a Gaussian random variable

$$T_n = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}, \quad \begin{aligned} X_i &\sim \mathcal{N}(\mu_{X_i}, \Sigma_{X_{i,i}}) \\ &= \mathcal{N}^{-1}(\eta_{X_i}, \Lambda_{X_{i,i}}) \\ T_n &\sim \mathcal{N}(\mu_n, \Sigma_n) \\ &= \mathcal{N}^{-1}(\eta_n, \Lambda_n) \end{aligned}, \tag{1}$$

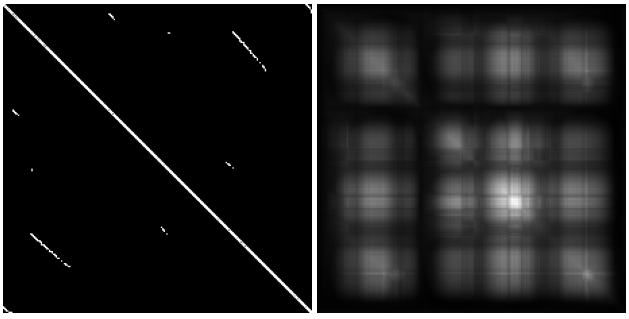

Figure 1. Comparison between a normalized information matrix (left) and the accompanying covariance matrix (right). Black pixels represent 0, while whiter the pixel the closer its value is to 1.

where μ_{X_i} and $\Sigma_{X_{i,i}}$ are mean and covariance of the state X_i, while μ_n and Σ_n are mean and covariance of the trajectory T_n, respectively. The equivalent representation of the Gaussian distribution in the information form is given by the relation $\eta = \Sigma^{-1}\mu$ and $\Lambda = \Sigma^{-1}$, thus, for example, the information vector and information matrix of of the state X_i are given by $\eta_{X_i} = \Sigma^{-1}_{X_{i,i}}\mu_{X_i}$ and $\Lambda_{X_{i,i}} = \Sigma^{-1}_{X_{i,i}}$, respectively. As shown in Eustice et al. (2006) the information matrix Λ_n of the ESDSF trajectory T_n has a sparse tridiagonal structure

$$\Lambda_n = \begin{bmatrix} \Lambda_{X_{n,n}} & \Lambda_{X_{n,n-1}} & \\ \Lambda_{X_{n-1,n}} & \Lambda_{X_{n-1,n-1}} & \Lambda_{X_{n-1,n-2}} \\ & \Lambda_{X_{n-2,n-1}} & \Lambda_{X_{n-2,n-2}} \\ & \vdots & \vdots \end{bmatrix}, \quad (2)$$

which is the result of using the motion model and trajectory augmentation equations described in the sequel. Sparsity of the information matrix is the key advantage of ESDSF, since it enables fast computation of the matrix inverse using specially designed sparse-matrix solvers. The illustration of difference between the normalized information matrix and the accompanying covariance matrix is shown in Fig. 1. These matrices were extracted from a real SLAM system at the end of one of the experiments presented in Sec. 7. The black pixels represent a value of 0, while the whiter the pixel is, the closer its value is to 1.

Each state X_i consists of a position and orientation that the robot had at the time when the state was added to the trajectory. State position and orientation are expressed in the coordinate frame assigned to the initial state X_1. Whenever a new state X_i is added to the trajectory, measurement z_i is taken from the sensor used to map the environment. In most modern SLAM systems, e.g., ORB-SLAM and LSD-SLAM, the term *keyframe* is used instead of the *state*. In our case, the term *state* is more appropriate since X_i belongs to the filter state space and its pose is directly estimated by the filter. However, since most modern SLAM solutions use graph optimization back-ends, the term *keyframe* is used instead of *state* to represent different nodes of the graph. In general both the *keyframe* and *state* represent the same and consist of: (i) the pose of the robot at the time they were added and (ii) measurement taken at the same moment.

While the robot moves, its current pose is estimated using a motion model described as the first order Markov process

$$X_{n+1} = f(X_n, \Omega_n, w_n), \quad (3)$$

where X_n represents the last state in the trajectory T_n, X_{n+1} represents the current robot pose, w_n represents zero-mean white Gaussian noise with covariance Q_n, while Ω_n stands for robot displacement between X_n and X_{n+1} obtained from odometry. Depending on the available sensors, this can be wheel odometry, visual odometry, laser odometry etc.

3.1 Prediction step

Whenever new odometry data becomes available and the current robot pose is estimated using (3), the prediction step of ESDSF is triggered. ESDSF prediction consists of two sub steps: (i) augmentation of the trajectory T_n with X_{n+1}, and, conditionally, (ii) marginalization of X_n that is subject to the pose difference between X_n and X_{n-1}.

3.1.1 Augmentation. The augmentation step always happens immediately after the current robot pose X_{n+1} is estimated using the motion model (3). During augmentation, trajectory T_n is augmented with X_{n+1} and thus becomes T_{n+1}. Before the augmentation of T_n, we can write the distribution of all the states in the information form as

$$\begin{aligned} &p(X_n, M|z_{1:n}, u_{1:k}) \\ &= \mathcal{N}^{-1}\left[\begin{pmatrix} \eta_{X_n} \\ \eta_M \end{pmatrix}, \begin{pmatrix} \Lambda_{X_nX_n} & \Lambda_{X_nM} \\ \Lambda_{MX_n} & \Lambda_M \end{pmatrix}\right], \end{aligned} \quad (4)$$

where $z_{1:n}$ denotes the history of all measurements, $u_{1:k}$ stands for history of all odometry data, and M represents all trajectory states except the last one, i.e., $M = \{X_1 \ldots ...X_{n-1}\}$. Note that usually there are more odometry data than the states in the trajectory, hence we use a different index. After the augmentation with the state X_{n+1}, using Markov first order assumption and the fact that poses and measurements are not correlated, we have:

$$\begin{aligned} &p(X_{n+1}, X_n, M|z_{1:n}, u_{1:k+1}) = \\ &= p(X_{n+1}|X_n, M, z_{1:n}, u_{1:k+1})\, p(X_n, M|z_{1:n}, u_{1:k+1}) \\ &= p(X_{n+1}|X_n, u_{1:k+1})\, p(X_n, M|z_{1:n}, u_{1:k}) \\ &= \mathcal{N}^{-1}(\eta_{n+1}, \Lambda_{n+1}). \end{aligned} \quad (5)$$

Final expression for Λ_{n+1} is

$$\Lambda_{n+1} = \begin{bmatrix} \boxed{Q_n^{-1}} & \boxed{-Q_n^{-1}F_n} & 0 \\ \boxed{-F_n^T Q_n^{-1}} & \Lambda_{X_{n,n}} + F_n^T Q_n^{-1} F & \Lambda_{X_n M} \\ 0 & \Lambda_{M X_n} & \Lambda_{MM} \end{bmatrix} \tag{6}$$

where F_n stands for the Jacobian of the motion model (3). The final expression for η_{n+1} is given in Appendix 9.1.1. From (6) we can se that augmenting the trajectory T_n with the new state X_{n+1} requires only the addition of three new blocks to the information matrix. All the other elements remain unchanged and sparsity is preserved.

When augmentation of the trajectory is complete, we check the pose difference between X_n and X_{n-1}. If the difference is larger than the predefined threshold, we conclude that the measurement associated with X_n provides new information and we keep both X_n and X_{n+1} within the state space and the prediction step ends. When new odometry data becomes available, the new robot pose is estimated based on X_{n+1} and the pose difference is then checked between X_{n+1} and X_n. Otherwise, we proceed with the marginalization step described in the sequel.

3.1.2 Marginalization. If the difference between X_n and X_{n-1} is smaller than the predefined threshold, state X_n is marginalized from the trajectory and replaced by X_{n+1}, i.e., $X_n \leftarrow X_{n+1}$. Marginalization of the state from the trajectory equals marginalizing a multivariate Gaussian distribution. In general, we can write the multivariate Gauss distribution as:

$$p\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}; \mu, \Sigma\right), \quad \begin{matrix} x \sim \mathcal{N}(\mu_x, \Sigma_{xx}) \\ y \sim \mathcal{N}(\mu_y, \Sigma_{yy}) \\ z \sim \mathcal{N}(\mu_z, \Sigma_{zz}) \end{matrix}, \tag{7}$$

where

$$\begin{aligned} \Sigma &= \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{xz} \\ \Sigma_{yx} & \Sigma_{yy} & \Sigma_{yz} \\ \Sigma_{zx} & \Sigma_{zy} & \Sigma_{zz} \end{bmatrix} = \Lambda^{-1} \\ &= \begin{bmatrix} \Lambda_{xx} & \Lambda_{xy} & \Lambda_{xz} \\ \Lambda_{yx} & \Lambda_{yy} & \Lambda_{yz} \\ \Lambda_{zx} & \Lambda_{zy} & \Lambda_{zz} \end{bmatrix}^{-1}. \end{aligned} \tag{8}$$

For example, if we want to marginalize y, we need to solve the following integral:

$$p(x,z) = \int p\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}; \mu, \Sigma\right) dy = \mathcal{N}^{-1}(\bar{\eta}, \bar{\Lambda}), \tag{9}$$

where $\bar{\eta}$ and $\bar{\Lambda}$ are equal to:

$$\bar{\eta} = \eta_\alpha - \Lambda_{\alpha\beta}\Lambda_\beta^{-1}\eta_\beta \tag{10}$$

$$\bar{\Lambda} = \Lambda_\alpha - \Lambda_{\alpha\beta}\Lambda_\beta^{-1}\Lambda_{\beta\alpha} \tag{11}$$

with

$$\begin{aligned} \eta_\alpha &= \begin{bmatrix} \eta_x \\ \eta_z \end{bmatrix}, \ \Lambda_\alpha = \begin{bmatrix} \Lambda_{xx} & \Lambda_{xz} \\ \Lambda_{zx} & \Lambda_{zz} \end{bmatrix}, \\ \Lambda_{\alpha\beta} &= \begin{bmatrix} \Lambda_{xy} \\ \Lambda_{zy} \end{bmatrix}, \ \Lambda_\beta = \Lambda_{yy}. \end{aligned} \tag{12}$$

Concretely, for the case of ESDSF we have to marginalize X_n from T_{n+1}:

$$\begin{aligned} &p(X_{n+1}, M | z_{1:n+1}, u_{1:k+1}) \\ &\quad = \int p(X_{n+1}, X_n, M | z_{1:n+1}, u_{1:k+1}) dX_n \\ &\quad = \mathcal{N}^{-1}(\bar{\eta}_n, \bar{\Lambda}_n). \end{aligned} \tag{13}$$

To find $\bar{\eta}_n$ and $\bar{\Lambda}_n$, we use (10) and (11), which yield for elements of (12) the following formulae:

$$\eta_\alpha = \begin{bmatrix} \eta_{X_{n+1}} \\ \eta_M \end{bmatrix}, \ \Lambda_\alpha = \begin{bmatrix} \Lambda_{X_{n+1,n+1}} & \Lambda_{X_{n+1}M} \\ \Lambda_{mX_{n+1}} & \Lambda_{MM} \end{bmatrix}, \tag{14}$$

$$\Lambda_{\alpha\beta} = \begin{bmatrix} \Lambda_{X_{n+1,n}} \\ \Lambda_{mX_n} \end{bmatrix}, \ \Lambda_\beta = \Lambda_{X_{n,n}}. \tag{15}$$

These equations can be further simplified, because after the trajectory augmentation with X_{n+1}, states X_n and X_{n+1} are only connected via neighboring states, which means that matrices $\Lambda_{X_{n+1}M}$ and $\Lambda_{mX_{n+1}}$ are zero matrices and only a single block in Λ_{mX_n} is different from zero. This yields the final expression for $\bar{\Lambda}_n$

$$\bar{\Lambda}_n = \begin{bmatrix} \alpha_n & Q_n^{-1}F_n\beta_n^{-1}\Lambda_{X_n M} \\ \Lambda_{MX_n}\beta_n^{-1}F_n^T Q_n^{-1} & \Lambda_{MM} - \Lambda_{MX_n}\beta_n^{-1}\Lambda_{X_n M} \end{bmatrix}, \tag{16}$$

where

$$\begin{aligned} \alpha_n &= (Q_n + F_n \Lambda_{X_{n,n}}^{-1} F_n^T)^{-1} \\ \beta_n &= (\Lambda_{X_{n,n}} + F_n^T Q_n^{-1} F_n). \end{aligned}$$

The final expression for $\bar{\eta}_n$ is given in Appendix 9.1.2. Taking into account that Λ_{mX_n} has only a single non-zero block, we can see from (16) that, similarly to augmentation, only four blocks of the information matrix Λ_n need to be changed during the marginalization. Once marginalization is complete, $\bar{\Lambda}_n$ and $\bar{\eta}_n$ become the new Λ_n and η_n, respectively, and $X_n \leftarrow X_{n+1}$, which then concludes the prediction step.

3.2 Update step

Update in the ESDSF SLAM is triggered every time the loop closing is detected between two trajectory states X_i and X_j. The measurement model in the ESDSF SLAM

system is given in the form of a relative pose between states X_i and X_j

$$y = h(X_i, X_j) + v,\ v \sim \mathcal{N}(0, R_{ij}), \tag{17}$$

where v represents measurement noise and is assumed to be a white zero-mean Gaussian with covariance matrix R_{ij}. The measurement is obtained from a relative pose estimation (RPE) algorithm based on the saved measurements z_i and z_j. The update equations of ESDSF are the same as EIF update equations. However, since y depends only on X_i and X_j, the measurement Jacobian H has a sparse structure

$$H_{n+1} = \begin{bmatrix} \cdots & 0 & \cdots & \frac{\partial h}{\partial X_i} & \cdots & \frac{\partial h}{\partial X_j} & \cdots \end{bmatrix}.$$

This means that the update always affects only four blocks in the information matrix that share information associated to X_i and X_j, thus making it constant time. Of course, after the update, vector μ_n has to be calculated as $\mu_n = \Lambda_n^{-1}\eta_n$, which is not constant time. However, this is drastically speed up due to the sparsity of the information matrix. For more details on the update step and the entire ESDSF please confer Eustice et al. (2006).

4 Lie group and algebra preliminaries

We now briefly introduce the necessary prerequisites for derivation of the ESDSF on Lie groups, while the interested reader can look for a more rigorous treatment of the subject in Chirikjian (2012). A Lie group G is a group which has the structure of a smooth manifold. A tangent space $T_X(\mathsf{G})$ is associated to $X \in \mathsf{G}$ such that it is placed at the group identity, called Lie algebra $\mathfrak{g}$, and then transferred to any $X \in \mathsf{G}$ by applying corresponding (left or right) action (Selig (2005)). The Lie algebra $\mathfrak{g}$ is an open neighbourhood around the zero-element in the tangent space of G at the identity. In this paper we use matrix Lie groups which are usually the ones considered in engineering and physical sciences.

The matrix exponential $\exp_{\mathsf{G}}$ and logarithm $\log_{\mathsf{G}}$ establish a local diffeomorphism between the group and the pertaining algebra

$$\exp_{\mathsf{G}} : \mathfrak{g} \to \mathsf{G} \ \text{ and } \ \log_{\mathsf{G}} : \mathsf{G} \to \mathfrak{g}\,. \tag{18}$$

The Lie algebra $\mathfrak{g} \subset \mathbb{R}^{n\times n}$ associated to a p-dimensional matrix Lie group $\mathsf{G} \subset \mathbb{R}^{n\times n}$ is a a p-dimensional vector space defined by a basis consisting of p real matrices E_r, $r = 1, .., p$, often referred to as generators, see Park et al. (2010). Furthermore, a natural relation between $\mathfrak{g}$ and $\mathbb{R}^p$ is given through a linear isomorphism by

$$[\cdot]_{\mathsf{G}}^{\vee} : \mathfrak{g} \to \mathbb{R}^p \ \text{ and } \ [\cdot]_{\mathsf{G}}^{\wedge} : \mathbb{R}^p \to \mathfrak{g}\,. \tag{19}$$

For brevity, we will use the following notation of Bourmaud et al. (2016)

$$\exp_{\mathsf{G}}^{\wedge}(x) = \exp_{\mathsf{G}}([x]_{\mathsf{G}}^{\wedge}) \ \text{ and } \ \log_{\mathsf{G}}^{\vee}(X) = [\log_{\mathsf{G}}(X)]_{\mathsf{G}}^{\vee}\,, \tag{20}$$

where $x \in \mathbb{R}^p$ and $X \in \mathsf{G}$. In addition, we need two more operators—the adjoint representation of a Lie group and Lie algebra, respectively denoted as Ad_{G} and ad_{G}. They appear due to general non-commutative nature of matrix Lie groups, i.e., $XY \neq YX$. However, the non-commutativity can be captured by the so-called adjoint representation of G on $\mathfrak{g}$ as follows

$$X \exp_{\mathsf{G}}^{\wedge}(y) = \exp_{\mathsf{G}}^{\wedge}(\mathrm{Ad}_{\mathsf{G}}(X)y)X, \tag{21}$$

where $X \in \mathsf{G}$, $y \in \mathbb{R}^p$. This can be seen as a way of representing the elements of the group as a linear transformation of the group's algebra. The adjoint representation of $\mathfrak{g}$, ad_{G}, is in fact the differential of Ad_{G} at the identity. A more detailed discussion on these concepts and the used notation can be found in Chirikjian (2012).

To make use of ESDSF on Lie groups, we first need to establish an error distribution on Lie groups. If a random variable describing the error, $\epsilon \triangleq \log_{\mathsf{G}}^{\vee}(X^I)$, is tightly focused around the identity element X^I, it can be well described with a Euclidean Gaussian in the pertaining algebra $\epsilon \sim \mathcal{N}_{\mathbb{R}^p}(\mathbf{0}^{p\times 1}, P)$ as in Wang and Chirikjian (2008), and then we say that X follows a concentrated Gaussian distribution (CGD) on G around the identity element (confer Bourmaud et al. (2015) for details). We regard this distribution then as a distribution on Lie groups at the identity element, but which can further be translated over G by using the left action. Finally, by combining the error distribution definition and left action, a random variable $X \in \mathsf{G}$ is defined as

$$X = \mu \exp_{\mathsf{G}}^{\wedge}(\epsilon)\ \text{, with } \ X \sim \mathcal{G}(\mu, P)\,, \tag{22}$$

with mean value $\mu \in \mathsf{G}$, covariance $P \in \mathbb{R}^{p\times p}$, and $\mathcal{G}$ denoting the CGD as in Wang and Chirikjian (2008). In the present paper, we concretely employ the special Euclidean group $\mathsf{SE}(3)$ and for the completeness of the paper, required group operators are given in Appendix 9.2.

5 ESDSF on Lie groups

To derive the ESDSF on Lie groups we need several building blocks. Fundamentally, we require the information form of the LG-EKF of Bourmaud et al. (2015), and we need to be able to compute the augmentation and marginalization of a CGD. In Ćesić et al. (2017) we solved the problem of the information form and proposed the extended information filter on Lie groups (LG-EIF). The augmentation and marginalization of the CGD were

presented in Bourmaud et al. (2016) in the context of iterated LG-EKF, where authors solved the prediction step by approximating the Chapman-Kolmogorov equation with a joint distribution and then marginalizing the posterior state. In this paper we propose to follow the same train of thought and we extend this result to the prediction equations of ESDSF on Lie groups. In the sequel, we derive the proposed LG-ESDSF.

5.1 State space construction

Let us represent a trajectory state X_i of the robot trajectory T_n by an $\mathsf{SE}(3)$ group element

$$X_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}, X_i \sim \mathcal{G}(\mu_{X_i}, \Sigma_{X_{i,i}}), \tag{23}$$

where R_i is a member of the special orthogonal group $\mathsf{SO}(3)$, given as a 3×3 rotation matrix defining robot orientation in the global frame, and $t_i = [x_i, y_i, z_i]$ represents the robot position in the global frame. In contrast to the Euclidean ESDSF, the trajectory T_n is no longer a vector, but rather a block diagonal matrix consisting of n $\mathsf{SE}(3)$ elements, i.e.,

$$T_n = \begin{bmatrix} X_1 & 0 & 0 & 0 \\ 0 & X_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & X_n \end{bmatrix} \in \mathsf{G} = \mathsf{SE}(3) \times ... \times \mathsf{SE}(3). \tag{24}$$

However, following the idea of the information filter approach to LG-EIF of Ćesić et al. (2017), rather than keeping the trajectory in the form of the matrix T_n, we store the states in the form of concatenated Lie algebra $\mathfrak{se}(3)$ elements

$$\begin{aligned} \tau_n = \log^{\vee}_{\mathsf{G}}(T_n) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \log^{\vee}_{\mathsf{G}}(X_1) \\ \log^{\vee}_{\mathsf{G}}(X_2) \\ \vdots \\ \log^{\vee}_{\mathsf{G}}(X_n) \end{bmatrix}, \\ x_i \sim \mathcal{N}(\mu_{x_i}, \Sigma_{x_{i,i}}) = \mathcal{N}^{-1}(\eta_{x_i}, \Lambda_{x_{i,i}}), \\ \tau_n \sim \mathcal{N}(\mu_n, \Sigma_n) = \mathcal{N}^{-1}(\eta_n, \Lambda_n), \end{aligned} \tag{25}$$

where $x_i \in \mathfrak{se}(3)$ represents the state $X_i \in \mathsf{SE}(3)$ mapped to the Lie algebra, while τ_n can be seen as a whole trajectory T_n mapped to the Lie algebra. The relations between the information matrix Λ_n, the information vector η_n, the trajectory mean value in Lie algebra μ_n, and the covariance matrix Σ_n follow the same equations as in the standard ESDSF, i.e., $\mu_n = \Lambda_n^{-1}\eta_n$ and $\eta_n = \Sigma_n^{-1}\mu_n$.

5.2 Motion model and LG-ESDSF prediction

The motion prediction is assumed to follow a non-linear first order Markov process similarly to (3), except that the motion is now defined directly on G as

$$X_{n+1} = f(X_n, \Omega_n, w_n) = X_n \exp^{\wedge}_{\mathsf{G}}(\Omega_n + w_n), \tag{26}$$

where $X_n \in \mathsf{G}$ is the state, $w_n \sim \mathcal{N}_{\mathbb{R}^p}(0, Q_n)$ is a p-dimensional white Gaussian process noise, and $\Omega_n = [\Delta t, \Delta r]$ represents a robot displacement measured by odometry between X_n and X_{n+1}. Change in the position component is represented by $\Delta t = [\Delta x, \Delta y, \Delta z]$, while the change in rotation Δr is represented using the Lie algebra parametrization of the special orthogonal rotation group $\mathsf{SO}(3)$ (Euler-axis convention). The process noise covariance Q_n represents uncertainty of the odometry. The state covariance matrix is propagated as follows

$$\Sigma_{n+1} = \mathcal{F}_n \Sigma_n \mathcal{F}_n^{\mathrm{T}} + \Psi(\Omega_n) Q_n \Psi(\Omega_n)^{\mathrm{T}} \tag{27}$$

$$\mathcal{F}_n = \mathrm{Ad}\left(\exp^{\wedge}_{\mathsf{G}}(-\Omega_n)\right) + \Psi(\Omega_n)\mathscr{C}_k, \tag{28}$$

where Ψ is the right Jacobian of G (see Barfoot and Furgale (2014)), while $\mathscr{C}_k$ denotes the linearization of the motion model (26) at X_n as in Bourmaud et al. (2015), which is given as

$$\Psi(v) = \sum_{m=0}^{\infty} \frac{(-1)^m}{(m+1)!} \mathrm{ad}(v)^m, \; v \in \mathbb{R}^p, \tag{29}$$

$$\mathscr{C}_n = \frac{\partial}{\partial \epsilon} \Omega\left(X_n \exp^{\wedge}_{\mathsf{G}}(\epsilon)\right)|_{\epsilon=0}. \tag{30}$$

The equation (27) looks similar to the EKF prediction, with the Jacobian-like matrix $\mathcal{F}_n$ resulting from the linearization of the motion model (26) and general non-commutativity of matrix Lie groups. The matrix Ψ is reparametrization resulting from dislocation over Lie group which appears in the prediction step. Since $\Omega(\cdot)$ is not a function of the state X_n, (30) evaluates to zero, i.e., $\mathscr{C}_n = 0$, hence (28) evaluates to $\mathcal{F}_n = \mathrm{Ad}\left(\exp^{\wedge}_{\mathsf{G}}(-\Omega_n)\right)$. For brevity, we also introduce the following notation

$$\mathcal{Q}_n = \Psi_n Q_n \Psi_n^{\mathrm{T}}, \quad \Psi_n = \Psi(\Omega_n). \tag{31}$$

In the prediction step of the LG-ESDSF, similarly as in the prediction step of the ESDSF, the trajectory is first augmented with the new state X_{n+1}. Afterwards, depending on the marginalization threshold, the state X_n can be either kept or marginalized. Given that, if the threshold is exceeded, the state X_n is permanently kept as a part of T_{n+1}. The distribution parameters Λ_{n+1} and η_{n+1}

associated to the trajectory τ_{n+1} evaluate to

$$\eta_{n+1} = \begin{bmatrix} \mathcal{Q}_n^{-1}(\mu_{x_{n+1}} - F\mu_{x_n}) \\ \eta_{x_n} - \mathcal{F}_n^T \mathcal{Q}_n^{-1}(\mu_{x_{n+1}} - \mu_{x_n}) \\ \eta_{x_{n-1}} \\ \vdots \end{bmatrix}, \tag{32}$$

$$\Lambda_{n+1} = \begin{bmatrix} \mathcal{Q}_n^{-1} & -Q_n^{-1}\mathcal{F}_n & 0 \\ -\mathcal{F}_n^T Q_n^{-1} & \Lambda_{x_n,n} + \mathcal{F}_n^T \mathcal{Q}_n^{-1}\mathcal{F}_n & \Lambda_{x_{n,n-1}} \\ 0 & \Lambda_{x_{n-1,n}} & \Lambda_{x_{n-1,n-1}} \\ \ddots & \ddots & \ddots \end{bmatrix}. \tag{33}$$

If the marginalization threshold is not exceeded and the marginalization needs to be performed, the previous state X_n is removed, while the new state X_{n+1} becomes X_n. The equations for combining augmentation and marginalization are in the vein of the iterated LG-EKF prediction presented in Bourmaud et al. (2016); however, for the information form and LG-ESDSF the procedure is different, and we present the resulting expressions for η_n and Λ_n

$$\eta_n = \begin{bmatrix} \mathcal{Q}_n^{-1}\mathcal{F}_n\beta_n^{-1}\eta_{x_n} + \alpha_n(\mu_{x_{n+1}} - \mathcal{F}_n\mu_{x_n}) \\ \eta_{x_{n-1}} - \Lambda_{x_{n-1}x_n}(\eta_{x_n} - \mathcal{F}_n^T\mathcal{Q}_n^{-1}(\mu_{x_{n+1}} - \mathcal{F}_n\mu_{x_n})) \\ \eta_{x_{n-2}} \\ \vdots \end{bmatrix} \tag{34}$$

$$\Lambda_n = \begin{bmatrix} \alpha_n & \mathcal{Q}_n^{-1}\mathcal{F}_n\beta_n^{-1}\Lambda_{x_{n,n-1}} & 0 \\ \Lambda_{x_{n-1,n}}\beta_n^{-1}\mathcal{F}_n^T\mathcal{Q}_n^{-1} & \gamma_n & \Lambda_{x_{n,n-1}} \\ 0 & \Lambda_{x_{n,n-1}} & \Lambda_{x_{n-1,n-1}} \\ \ddots & \ddots & \ddots \end{bmatrix}, \tag{35}$$

where

$$\begin{aligned} \alpha_n &= (\mathcal{Q}_n + \mathcal{F}_n\Lambda_{x_n,n}^{-1}\mathcal{F}_n^T)^{-1}, \\ \beta_n &= (\Lambda_{x_n,n} + \mathcal{F}_n^T\mathcal{Q}_n^{-1}\mathcal{F}_n), \\ \gamma_n &= \Lambda_{x_{n-1,n-1}} - \Lambda_{x_{n-1,n}}\beta_n^{-1}\Lambda_{x_{n,n-1}}. \end{aligned} \tag{36}$$

Given that, we have derived the prediction step of the LG-ESDSF.

5.3 Measurement model and LG-ESDSF update

The discrete measurement model on matrix Lie groups is given as

$$Z_{n+1} = h(T_{n+1})\exp_{\mathsf{G}'}^{\wedge}(v_{n+1}), \tag{37}$$

where $Z_{n+1} \in \mathsf{G}'$, $h : \mathsf{G} \to \mathsf{G}'$ is a $\mathcal{C}^1$ function, G' is a q-dimensional Lie group and $v_{n+1} \sim \mathcal{N}_{\mathbb{R}^q}(\mathbf{0}^{q\times 1}, R_{n+1})$ is zero-mean white Gaussian noise with covariance R_{n+1}.

The update step in LG-SLAM occurs following similar rationale as in the case of the Euclidean ESDSF SLAM, i.e., it is performed whenever a loop closing between any two states X_i and X_j is detected, and a relative pose measurement is delivered. However, in reality, update will almost always occur between the newly augmented state X_n and one or more states already in the trajectory. For this reason, in the remainder of this section, we assume that the trajectory T_{n+1} is being updated after the loop closing occurred between X_n and X_j, $0 < j < n$. No generality is lost since all equations are valid if we substitute X_n with X_i ($i \neq j$, $1 < i \leq n$). Hence, the filter uses the relative poses of the RPE module as measurements. The measurement function is given as

$$h(T_{n+1}) = X_j^{-1}X_n \in \mathsf{SE}(3), \tag{38}$$

which represents the relative transformation between poses X_j and X_n. The innovation term of the LG-ESDSF SLAM is modelled by the difference in relative poses between the measurement function (38) and the RPE module providing measurement in term of relative transformation in loop closing. It is finally given as

$$z_{n+1} = \log_{\mathsf{G}}^{\vee}\left(h\left(T_{n+1}\right)^{-1} Z_{n+1}\right), \tag{39}$$

where z is already an innovation vector since $\log_{\mathsf{G}}^{\vee}$ is applied. For calculating the updated estimates of the information matrix Λ_{n+1}^{+} and the information vector η_{n+1}^{+}, we use LG-EIF update equations from Ćesić et al. (2017). We begin by calculating Λ_{n+1}^{-} and η_{n+1}^{-} using standard EIF equations

$$\begin{aligned} \eta_{n+1}^{-} &= \mathcal{H}_{n+1}^{\mathrm{T}} R_{n+1}^{-1} z_{n+1}, \\ \Lambda_{n+1}^{-} &= \Lambda_{n+1} + \mathcal{H}_{n+1}^{\mathrm{T}} R_{n+1}^{-1} \mathcal{H}_{n+1}, \end{aligned} \tag{40}$$

with R_{n+1} being the measurement uncertainty reported by RPE and matrix $\mathcal{H}_{n+1}$ is evaluated as in Bourmaud et al. (2015)

$$\mathcal{H}_{n+1} = \frac{\partial}{\partial \epsilon}\left[\log_{\mathsf{G}}^{\vee}\left(h(T_{n+1})^{-1}h\big(T_{n+1}\exp_{\mathsf{G}}^{\wedge}(\epsilon)\big)\right)\right]\Big|_{\epsilon=0}. \tag{41}$$

For the $\mathsf{SE}(3)$ group calculating the relative pose between states X_n and X_j reduces to simple matrix inverse and multiplication. Given the measurement model (38), the matrix (41) evaluates to

$$\mathcal{H}_{n+1} = [\underbrace{0\cdots}_{1:j-1 \text{ zero bl.}} \overbrace{-\operatorname{Ad}(X_{n+1}^{-1})\operatorname{Ad}(X_j)}^{j\text{th block}} \underbrace{\cdots 0\cdots}_{j+1:n-1 \text{ zero bl.}} \overbrace{I}^{n\text{th bl.}} 0].$$

We note that a similar result was obtained in Bourmaud et al. (2016) for relative pose averaging, although without

Algorithm 1 LG-ESDSF SLAM back-end pseudocode

Get current robot pose X_{n+1} § 5.2
Get odometry data Ω_n
Perform motion model (26) to get X_{n+1}
ESDSF prediction § 5.2
Calculate matrices $\mathcal{F}_n$ (28) and $\mathcal{Q}_n$ (31)
AUGMENT state X_{n+1}
Calculate η_{n+1} (32) and Λ_{n+1} (33)
if MARGINALIZE state X_n **then**
Calculate η_n (34) and Λ_n (35)
else if LOOP CLOSED between X_n and X_j **then**
ESDSF update § 5.3
Get measurement Z_{n+1} from RPE
Calculate innovation z_{n+1} (39)
LG-EIF update
Calculate η^-_{n+1} and Λ^-_{n+1} (40)
CGD reparametrization
Calculate η^+_{n+1} (44) and Λ^+_{n+1} (43)
end ESDSF update
end if
end ESDSF prediction

derivation, which is why we provide it for completeness in Appendix 9.3. This result shows that, similarly to the Euclidean ESDSF, the matrix $\mathcal{H}_{n+1}$ remains sparse consisting of n 6×6 blocks, among which only blocks j and n are non-zero. However, as explained in Ćesić et al. (2017) for LG-EIF, η^-_{n+1} and Λ^-_{n+1} (denoted with superscript $^-$) do not represent final estimates of updated information matrix and information vector, since at this point the mean value $\mu^-_{n+1} = (\Lambda^-_{n+1})^{-1}\eta^-_{n+1}$ is in general a non-zero vector, thus in collision with the CGD definition (22). To overcome this issue, the state reparametrization (denoted with superscript $^+$) is performed as proposed in Bourmaud et al. (2015), and the final formulae are given as follows

$$\mu^-_{n+1} = (\Lambda^-_{n+1})^{-1}\eta^-_{n+1} \tag{42}$$

$$\Lambda^+_{n+1} = \Psi(\mu^-_{n+1})^{-T}\Lambda^-_{n+1}\Psi(\mu^-_{n+1})^{-1} \tag{43}$$

$$\eta^+_{n+1} = \Lambda^+_{n+1}\log^\vee_{\mathsf{G}}\left(\exp^\wedge_{\mathsf{G}}(\Lambda_n^{-1}\eta_n)\exp^\wedge_{\mathsf{G}}(\mu^-_n)\right). \tag{44}$$

Note that η^+_{n+1} and Λ^+_{n+1} differ from η_{n+1} and Λ_{n+1} which are obtained only via augmentation in the prediction step. Given that, we have derived the update step of the LG-ESDSF. Pseudocode of the entire LG-ESDSF SLAM back-end is given in Algorithm 1 while LG-SLAM back-end coupled with a SLAM front-end is shown in Fig. 2.

5.4 Covariance estimation and computational complexity analysis

In order to perform both the prediction and update, LG-SLAM uses relative pose estimated between different time instances. The prediction step relies on the relative pose between two consecutive states determined by the odometry module and used in the motion model (26), while the update step uses the relative pose evaluated by the RPE module between the two states for which the loop closing is detected in order to calculate innovation term (39). To incorporate these measurements, LG-SLAM needs information about their uncertainties, where Q_n represents the odometry uncertainty used in the prediction (27) , while R_n represents uncertainty of the RPE algorithm used in the update (40). The proposed filter operates in such a way that by the CGD definition (22), the uncertainties are assigned to variables located in the Lie algebra. For example, since algebra of the special orthogonal group is given in the form of the Euler axis representation, the uncertainties have to be associated to the same variable type. Since majority of sensors and relative pose estimation algorithms rely on Euler angle representation and associate the uncertainties in this representation, we apply the unscented transform (UT) presented in Julier and Uhlmann (2004), on the covariance matrices, in order for them to become adequately associated to the Lie algebra variables. The UT algorithm first takes measurement and the associated covariance matrix in Euler angles as input, and then applies the nonlinear transformation function resulting with the Euler axis representation and the new covariance matrix.

Considering the performance of LG-ESDSF as a SLAM back-end, we need to look at the computation complexity of each step. The prediction step of LG-ESDSF is constant-time similar to ESDSF, since it always changes the same number of blocks in the information matrix. In the case when the augmentation is not followed by marginalization, three new blocks are added (33), while in the case when marginalization is performed, only four existing blocks are changed (35). However, in both cases μ_{x_n} and $\mu_{x_{n+1}}$, residing in the Lie algebra $\mathfrak{se}(3)$, are necessary to complete the calculation of the new information vector η_{n+1} (32). Value $\mu_{x_{n+1}}$ can easily be calculated from $\mu_{X_{n+1}}$, but μ_{x_n} would have to be extracted from the μ_n which represents the expected value of trajectory τ_n (25), and requires computing $\mu_n = \Lambda_n^{-1}\eta_n$; moreover, new μ_n needs to be evaluated after every update. This is a computationally demanding operation, due to the inversion of the information matrix, and should be avoided. Since after the update, states permanently added to the trajectory do not change until the next update, we can construct an auxiliary vector μ'_n which stores $\mathfrak{se}(3)$ states permanently added between the updates. After the update is performed, we set $\mu'_n = \mu_n$ and continue by adding the new states in μ'_n until the next update. This way we always have the history of all the states, which can then be exploited in the prediction step with no need for inversion. Although in the prediction we need only μ_{x_n}, the update step uses

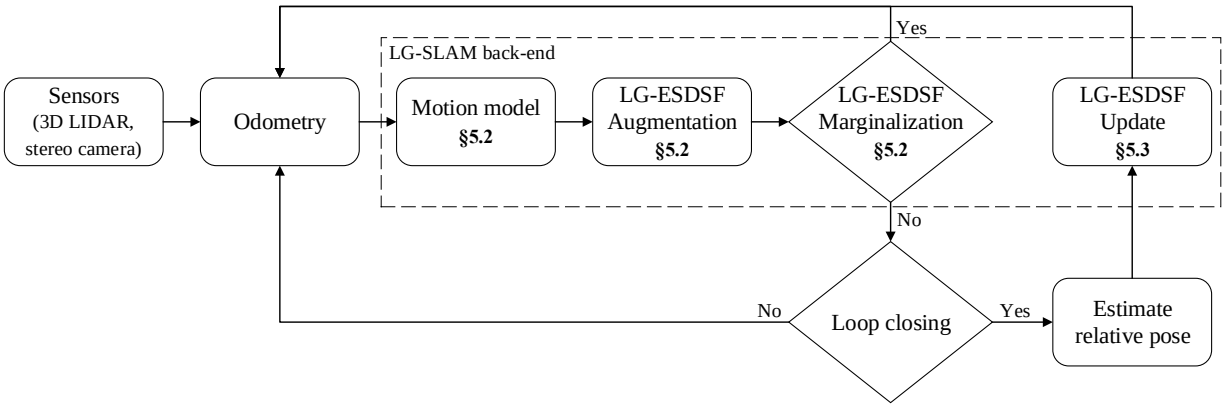

Figure 2. Schematic layout of the proposed LG-SLAM system.

the entire trajectory μ'_n. After loop closing between states X_i and X_j is detected, we require their respective poses in order to perform the update, meaning that we would need to calculate $\mu_n = \Lambda_n^{-1}\eta_n$ and extract μ_{x_i} and μ_{x_j}. To avoid the calculation of μ_n, we simply obtain μ_{x_i} and μ_{x_j} from the auxiliary vector μ'_n. Since we cannot know upfront which states will be included in the update, we have to keep the history of all the states.

By examining equations (42) to (44) it would appear that there is an extra inversion of Λ_{n+1} with respect to ESDSF. In ESDSF the inversion only needs to be computed after the update is performed, in order to recover the new μ_{n+1}, while in LG-ESDSF it is also required in (42). To solve this issue we rewrite (44) as follows

$$\begin{aligned}\eta^+_{n+1} &= \Lambda^+_{n+1}\log^\vee_{\mathsf{G}}\left(\exp^\wedge_{\mathsf{G}}(\Lambda_n^{-1}\eta_n)\exp^\wedge_{\mathsf{G}}(\mu^-_{n+1})\right) \\ &= \Lambda^+_{n+1}\log^\vee_{\mathsf{G}}\left(\exp^\wedge_{\mathsf{G}}(\mu_n)\exp^\wedge_{\mathsf{G}}(\mu^-_{n+1})\right) \\ &= \Lambda^+_{n+1}\mu^+_{n+1}\,,\end{aligned} \tag{45}$$

which means that there is no need for final computation of μ_{n+1} as it is already calculated within η^+_{n+1}. Therefore, there is also only a single inversion of the information matrix in LG-ESDSF.

The last potentially time consuming calculation is the inversion of $\Psi(\mu^-_{n+1})$ required in (43) during the update. Although $\Psi(\mu^-_{n+1})$ does have the same dimension as Λ^-_{n+1}, it is also a sparse matrix and keeps a strictly tridiagonal block form (not effected by the update); hence, its inversion reduces to n inversions of a 6×6 matrix.

6 Long-term LG-SLAM

In general, there are two scenarios for which we can consider the problem of long-term SLAM:

1. The robot explores new areas for a longer period of time.
2. The robot continuously moves inside the same area for a longer period of time.

For both of these scenarios and regardless of the used back-end and front-end, operation time of any SLAM system is limited. This is because the number of features in the map and/or states in the trajectory constantly rises and increases memory and computation requirements. In order to ensure long-term operation, number of states and features has to be reduced continuously. For the first scenario, information from the SLAM front-end is essential for reducing the size of the state-space regardless on the back-end. This is why, in this paper, we focus on solving the second scenario using only the information available to the pose graph SLAM back-end.

When the robot continuously moves through the same environment and if the environment being explored is reasonable in size, most pose graph SLAM systems will be able to build a complete map of the environment. The problems will occur when the robot continues to move repeatedly through the same environment, since the state space will continue to grow. The easiest solution would be to stop the SLAM system once the map has been built and use the constructed map to localize the robot within. Although straightforward, the main drawbacks of this solution are that (i) the robot cannot explore new areas and then return to the explored part without reinitializing the SLAM, and (ii) the robot can no longer improve map accuracy by closing loops. A better solution would be to allow SLAM to function continuously, but manage the increase in the number of features and/or trajectory states.

We build our solution to this problem under the assumption that when the robot moves through an already explored environment, majority of the newly augmented states will have similar poses with the already existing states in the trajectory. These states have to be added into the trajectory such that loop closing can occur and trajectory update can be performed. However, once the trajectory update is completed, measurements assigned to these states hold little new information. Given that, we

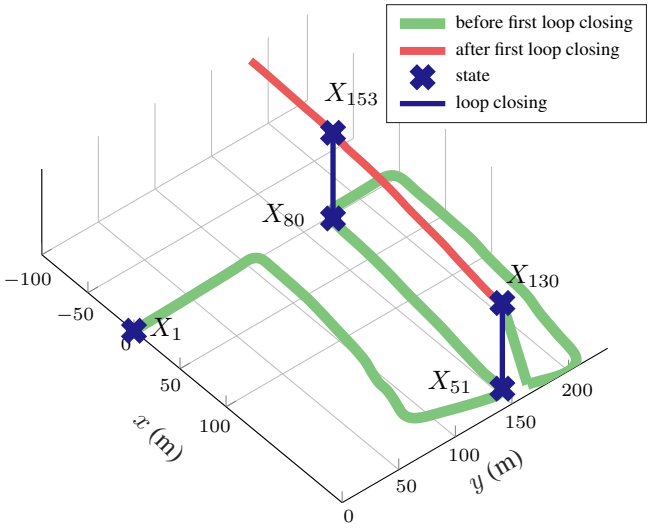

Figure 3. Example of a robot trajectory suitable for removing some already added states. Robot started moving from state X_1. First loop closing occurred between states (X_{130}, X_{51}) and second loop closing occurred between states (X_{153}, X_{80}).

formulate our main goal: *After each trajectory update, all the states having similar pose to some other states in the trajectory, added permanently after the previous update had occurred, should be removed (marginalized) in order to preserve long-term real-time operation*. Given that, we have to marginalize states after the update in order to allow the update to occur in the first place. Moreover, marginalizing states immediately after the update occurs ensures that accumulated odometry error is corrected. If we would marginalize them prior to the update, due to the odometry error, we could have states falsely further/closer from/to each other. By removing a state falsely detected as close, we lose information. On the other hand, if we do not remove truly close states, we retain redundant information.

The drawback of marginalizing states based only on their closeness is that the information about the dynamic changes in the environment may be lost. One solution to this problem would be to estimate the number of duplicate map landmarks between measurements assigned to two close states and then marginalize the state only if the number of duplicates exceeds a predefined threshold. However, this approach would require information from the front-end, which is in contrast to our goal of developing a back-end solution that is highly independent on the front-end selection, but that can easily be fitted to one, once a particular front-end is selected.

6.1 State selection and marginalization

In this section we assert that we can safely remove a state X_i from the trajectory, only if it was added between two consecutive loop closings. For example, let us consider a robot moving as shown in Fig. 3. Robot started in the initial state X_1 and continued adding states to the trajectory until the state X_{130} was added. Then, the loop was closed between X_{51} and X_{130} and the trajectory was updated. Afterwards, the robot continued moving until state X_{153} was added and loop was closed with X_{80}. The difference in height between the green and red path exists only for illustration purposes, in reality both paths lie practically on the same plane.

Now lets determine which states between X_1 and X_{153} should be marginalized. Since the loop closings occurred in states X_{130} and X_{153} we can assert that trajectory between them is accurate and we can, with high certainty, determine which states are close. From Fig. 3 we can see that all the states between X_{130} and X_{153} have similar poses with the states added between X_{51} and X_{80}. This means that most of the measurement information contained within these states is already contained within measurements assigned to states from X_{51} to X_{80}. This means that states from X_{130} to X_{153} can be safely marginalized.

In order to determine how similar the poses of two states X_i and X_j are (i.e., how close they are) we use the function f_c, which essentially takes into account Euclidean distance and orientation difference between two $\mathsf{SE}(3)$ elements.

$$f_c(X_i, X_j) = e_{\text{tran}} + \kappa e_{\text{rot}}, \tag{46}$$

where e_{tran} represents Euclidean distance between states X_i and X_j, e_{rot} represents difference in their orientation and κ is a scaling factor allowing for flexibility in weighting the contribution of the translation and rotation component. In order to evaluate e_{tran} and e_{rot} we first calculate relative pose between X_i and X_j

$$\Delta P = X_i^{-1} X_j = \begin{bmatrix} & R & & \Delta x \\ & & & \Delta y \\ & & & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{47}$$

and then evaluate their Euclidean distance as

$$e_{\text{tran}} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{48}$$

and difference between their orientations as

$$e_{\text{rot}} = \arccos\left(\frac{\text{trace}(R) - 1}{2}\right). \tag{49}$$

Now let us designate with β the set of all states between X_{130} and X_{153}, from Fig. 3, that can be removed according to (46). Let's assume that, without the loss of generality, all states X_i, $130 < i < 153$ are close enough to some previously added state X_j, $51 < j < 80$ so they can all be marginalized. The question arises, how to remove these states from the trajectory and from the information matrix? The most important fact about marginalizing states added

between two loop closings is that they are *connected only with the neighbouring states*. Given that, we can use (10) and (11), which we used to marginalize X_n during the LG-ESDSF prediction step, and the sparsity of the information matrix will remain preserved. The only changes in the information matrix after the removal of states β, besides size reduction, will be changes to the blocks related to states X_{130} and X_{153}. Moreover, since states in β are neighbouring states, they can be marginalized in a single block which increases computation speed in comparison to marginalizing just each state separately.

After successfully removing states in β, we have to decide what to do with X_{130} and X_{153}. The simplest solution would be to never marginalize states included in loop closings. However, in the case of the robot continuously moving through the same environment, we can expect a large number of loop closings; thus, continuously ignoring these states would inevitably result in the increase of the information matrix size. Therefore, we also have to marginalize states included in loop closings in order to ensure the long-term capability of our SLAM. However, due to the nature of the loop closing procedure, we only need to marginalize the state in which previous loop closing occurred, since the state, X_{153}, will become in the next iteration the previous loop closing state, as state X_{130} is currently. This is why in this example we would only need to marginalize X_{130}. The problem with removing loop closing state is that, as mentioned before, (10) and (11) should only be used if the state to be marginalized is connected only with its neighbouring states. However, this is never the case for states that participated in loop closings. If we did simply use (10) and (11) for marginalization, as explained in the next section, this would have a negative effect on the sparsity of the information matrix. Possible solutions are to use an approach that preserves the sparsity and reduces the number of loop closings. In LG-SLAM we have implemented both.

6.2 Marginalization of states included in the loop closings

For the simplicity of equations, in this section we assume that $X_l = X_{130}$. In order to understand why marginalization of the state X_l would have a negative impact on the sparsity of the information matrix, let us assume that we have already marginalized all the states in β. If we were to use (10) and (11) to marginalize X_l, we would obtain:

$$\Lambda = \begin{bmatrix} \Lambda_\alpha - \Lambda_{\alpha l}\Lambda_l^{-1}\Lambda_{l\alpha} & \Lambda_{\alpha\gamma} - \Lambda_{\alpha l}\Lambda_l^{-1}\Lambda_{l\gamma} \\ \Lambda_{\gamma\alpha} - \Lambda_{\gamma l}\Lambda_l^{-1}\Lambda_{l\alpha} & \Lambda_\gamma - \Lambda_{\gamma l}\Lambda_l^{-1}\Lambda_{l\gamma} \end{bmatrix}, \quad (50)$$

$$\Lambda_{\alpha l} = \begin{bmatrix} \dots \Lambda_{l,51} \dots \Lambda_{l,l-1} \end{bmatrix}^T, \; \Lambda_{\gamma l} = \begin{bmatrix} \Lambda_{153,l} \\ 0 \end{bmatrix}, \quad (51)$$

where Λ_α represents information from the states $\alpha = \{X_1, X_1 \dots X_{l-1}\}$ augmented before the state X_l, Λ_γ represents information of states $\gamma = \{X_{153}, X_{154}\}$ and $\Lambda_{\alpha l}$, $\Lambda_{\alpha\gamma}$, and $\Lambda_{l\gamma}$ represent their cross-information. As the result of the following expression:

$$\Lambda_\alpha - \Lambda_{\alpha l}\Lambda_l^{-1}\Lambda_{l\alpha}, \quad (52)$$

two blocks, $\Lambda_{l-1,51}$ and $\Lambda_{51,l-1}$, will be inserted into the new information matrix, while four more blocks will be added by the following expressions:

$$\Lambda_{\alpha\gamma} - \Lambda_{\alpha l}\Lambda_l^{-1}\Lambda_{l\gamma}, \quad (53)$$

$$\Lambda_{\gamma\alpha} - \Lambda_{\gamma l}\Lambda_l^{-1}\Lambda_{l\alpha}. \quad (54)$$

In total, six new blocks will be inserted and seven existing blocks will be removed. While this may seem fine, the problems will start to occur during future marginalization. For example, after the third loop closing and marginalization of new β states, X_{153} is marginalized from the information matrix. The term $\Lambda_{\alpha,153}$ then becomes

$$\Lambda_{\alpha,153} = \begin{bmatrix} \dots \Lambda_{153,51} \dots \Lambda_{l,80} \dots \Lambda_{l,l-1} \end{bmatrix}^T. \quad (55)$$

This would result in five more blocks from (52) and four more block from (54) and (53). Although (54) and (53) always add four blocks, number of new blocks added by (52) will continue to increase and the number of removed blocks will always remain seven. If we would continue this process, the number of new blocks in matrix Λ would increase quadratically with the number of consecutively marginalized states that participated in loop closings. This describes the worse case scenario in which all the states from β are marginalized. If some states between consecutive loop closings are not in β and remain in the information matrix, or if we periodically choose not to marginalize loop closing states, the loss of sparsity would be slowed down. However, in that case we would still increase the number of states.

In order to solve the problem of losing sparsity of the information matrix, we need to find an approach to approximate the dense information matrix with a sparse one. Two most recent approaches applicable to our problem are the works presented in Kretzschmar and Stachniss (2012) and Carlevaris-Bianco et al. (2014). In Kretzschmar and Stachniss (2012) authors presented a solution for long-term SLAM by approximating the dense sparse matrix with a sparse one using the Chow Liu tree (CLT) of Chow and Liu (1968). CLT approximates a probability distribution $p(\theta)$ with $p'(\theta)$ in a way that (i) each variable is conditioned only on one other variable and that (ii) Kullback-Leibler

divergence between p and p' is minimized:

$$p(\theta) = p(\theta_m) \prod_{i=1}^{m-1} p(\theta_i|\theta_{i+1}, \ \dots \ , \theta_m) \tag{56}$$

$$\approx p(\theta_m) \sum_{i=1}^{m-1} p(\theta_i|\theta_{i+1}) = p'(\theta). \tag{57}$$

The authors used CLT to approximate only the elimination cliques disregarding constraints in the remainder of the pose graph. The authors then use the resulting CLT to compute the constraints which remain in the pose graph. In Carlevaris-Bianco et al. (2014) the authors compute CLT from the entire information matrix which ensures taking into account all the pose graph constraints. Furthermore, the authors also introduced generic linear constraint (GLC) factors to ensure that the resulting information matrix will have full-rank. In LG-SLAM we use the same sparsification method as of Carlevaris-Bianco et al. (2014), but we do not use GLC factors as we assume that all measurements are such that the full-rank of the information matrix will always be preserved.

By using the aforementioned sparsification method, we can ensure sparsity of the information matrix regardless of the states we chose to marginalize. However, it should be noted that with every approximation we lose some information and increase error in the SLAM trajectory. Moreover, sparsification creates some computational overhead and increases the update complexity. Given that, we do not perform sparsification after every marginalization of β, but only when the information matrix becomes so dense that its inversion takes too long to be acceptable for real-time operation.

6.3 Rejecting unnecessary loop closings

Although updating trajectory after each loop closing increases SLAM accuracy, closing larger loops affects accuracy more than does closing the smaller ones. An example of such a situation is depicted in Fig. 3. Update after the second loop closing in the state X_l will have a much larger impact on the overall accuracy, than update after the third loop closing in the state X_p, since more information is gained. Furthermore, there are also some negative effects of performing update after every loop closing: (i) the trajectory update always takes some time, (ii) sparsification is required more frequently, and (iii) there will be less states in β that can be marginalized faster. This is why we need to choose which loop closings are important and which can be rejected using a measure of the total information gained by the loop closing. For this purpose, we use approach similar to the one proposed in Stachniss et al. (2004). The idea is to build a graph ${}^{T}g$ from the information matrix. Each node of the graph represents a single state from the trajectory and the nodes are only connected if they represent neighboring states or if the loop was closed. Weight of each connection is calculated using (46).

Once the graph is generated, we can calculate information gain from closing the loop between states X_i and X_j by finding the shortest path from a node N_i to a node N_j using the A^* algorithm, and then calculating the total weight of the path. When calculated in such a way, the total weight of the path is also refereed to as the topological distance ${}^{T}d_{i,j}$ between states X_i and X_j. The higher ${}^{T}d_{i,j}$ the more information is gained from the loop closing. We can again refer to Fig. 3 for an example. The information gained from the first loop closing (X_{130}, X_{51}) is:

$${}^{T}d_{51,130} = \sum_{e=52}^{130} f_c(e-1, e),$$

while the information gained from the second loop (X_{153}, X_{80}) is:

$${}^{T}d_{o,p} = \sum_{e=52}^{80} f_c(e-1, e) + \mathbf{f_c}(\mathbf{51}, \mathbf{130}) + \sum_{e=130}^{152} f_c(e, e+1).$$

If the update did not occur between $(X_l, \ X_u)$, ${}^{T}d_{80,153}$ would have the following value

$${}^{T}d'_{80,153} = \sum_{e=81}^{130} f_c(e-1, e) + \sum_{e=130}^{152} f_c(e, e+1).$$

We can see that ${}^{T}d_{80,153}$ is much smaller than ${}^{T}d'_{80,153}$ since

$$\sum_{e=52}^{80} f_c(e-1, e) + \mathbf{f_c}(\mathbf{51}, \mathbf{130}) \ll \sum_{e=81}^{130} f_c(e-1, e).$$

Now that we have the measure of information gain for each loop closing we can use it to decide which loops to use for the update and which to ignore. By changing the frequency of accepted loop closings we can decide how often do we need to perform sparsification. If we accept frequent repeated loop closings, we waste time on computing trajectory update and we need to perform sparsification more often, while at the same time we gain little information. By reducing the number of accepted loop closings we allow more states to be marginalized out faster which, together with avoiding the calculation of trajectory update, increases the algorithm speed. Of course we cannot reject too many consecutive loop closings, as we risk increasing the trajectory error beyond allowable tolerances. The complete algorithm for reducing the number of states in LG-SLAM is summarized in Algorithm 2.

Note that the algorithm could be even more efficient if information from the front-end was used. But as stated

Algorithm 2 State number reduction

Last update occurred after loop closed between X_a, X_i
if Loop closed between X_b, X_j $(b > j)$ **then**
  Compute topological distance ${}^T d_{j,b}$ § 6.3
  if ${}^T d_{j,b} \geq {}^T d_{min}$ **then**
    Perform ESDSF update § 5.3
    For states between X_a and X_b find f_c^m (46) § 6.1
      $f_c^m = \min(f_c(m,n),\ 0 < n < a),\ a \leq m < b$
    Put states satisfying $f_c^m \leq f_c^{max}$ in set β § 6.1
    Put neighbouring states from β into blocks § 6.1
    Marginalize blocks using (10) and (11)
    Set $X_a = X_b$
    If required, perform sparsification § 6.2
  end if
end if
${}^T d_{min}$ and f_c^{max} are predefined thresholds

in the introduction of this section, our goal was to develop every component of the LG-ESDSF back-end independent on the front-end. However, the developed long-term algorithm can easily be fitted to a selected front-end by changing the value of the minimum topological distance ${}^T d_{min}$ and by using a different criterion to add states to be marginalized between two loop closings in the set β.

7 Experimental results

We have divided the experimental testing of the proposed LG-SLAM in three different scenarios. First, we compare LG-SLAM with two state-of-the-art visual SLAM algorithms, namely ORB-SLAM and LSD-SLAM, which use g^{2}o as the back-end. Second, we compare LG-SLAM and g^{2}o back-ends by using two different front-ends: (i) stereo odometry with feature tracking (SOFT) of Cvišić and Petrović (2015), and (ii) three-dimensional normal distributions transform (3D-NDT) of Stoyanov et al. (2012). Finally, we test the LG-SLAM long-term capability by making it work continuously in the same environment. All algorithms were implemented using the C++ programming language. Testing machine was a computer with Intel Core i7@2.6 Ghz processor and 8 GB of RAM. For solving the sparse matrix equations we used the Eigen library by Guennebaud et al. (2010).

Testing was conducted using two public datasets, the KITTI vision benchmark suite by Geiger et al. (2012) and the EuRoC dataset by Burri et al. (2016). The KITTI dataset consists of 22 sequences recorded on different routes under different conditions. Ground truth is provided only for the first 11 sequences while others are used for online evaluation. The dataset offers measurements acquired by 3D LIDAR Velodyne HDL-64E, 2 grayscale cameras Point Grey Flea 2 in stereo configuration and two color cameras Point Grey Flea 2 also in stereo configuration. Ground truth is provided by an accurate inertial navigation system OXTS RT 3003. For the recording of the dataset all sensors were mounted on a commercially available vehicle Volkswagen Passat. The EuRoC dataset contains in total 11 sequences, out of which five were recorded in a large machine hall and six were recorded in a so-called Vicon room (i.e., a room equipped with the Vicon motion capture system). For every sequence, measurements were recorded using the visual inertial (VI) sensor (Nikolic et al. (2014)) mounted on a hexacopter UAV AscTec Neo. The VI sensor provides stereo images and synchronized data from the inertial measurement unit (IMU). Depending on available texture, brightness, and UAV dynamics each sequence is labelled as easy, medium or difficult. Ground truth is provided by the Vicon motion capture system and a laser tracking system, depending on the environment.

Although the KITTI dataset provides its own metric, it does not evaluate absolute errors between the ground truth and the estimated results, but rather compares errors on parts of sequences that are from 100 to 800 metres long, hence the benefits obtained from loop closing only marginally affect the metric. As such, it is designed primarily for evaluating pure odometry rather than complete SLAM systems. Because of this we used the automatic evaluation tool developed for the EuRoC dataset available online[1], which relies on evaluation of the absolute error. However, it first tries to find the best fit between the tested and ground truth trajectories and then computes the error. This is why we also included a metric based on (46) and evaluated rotational e_{rot} and translation e_{trans} errors separately without any fitting. We than calculated the root-mean-squared-error (RMSE) and provided it with the rest of the results. Herein, the error calculated using the online tool is referred to as e_F, while RMSE of the absolute translational error and absolute rotational error are referred to as e_{trans} and e_{rot}, respectively. Since in the KITTI dataset only tracks 0, 2, 5, 6, 7, and 9 provide suitable loop closures for SLAM front-end based on stereo, we provide results only for these tracks.

7.1 Experimental comparison of LG-SLAM, ORB-SLAM, and LSD-SLAM

Although the stereo version of LSD-SLAM is not available as open source, the results of testing LSD-SLAM, as well as ORB-SLAM, on the KITTI and EuRoC datasets are available in their respective papers. Since we could not calculate e_{trans} and e_{rot}, we provide only e_F in Tables 1 and 2 for all solutions if available (results for LSD-SLAM are available for only three out of 11 EuRoC sequences while both ORB-SLAM and SOFT failed to produce meaningful result for the final EuRoC track, probably because of incorrect calibration parameters). We also provide results

Table 1. Results of LG-SLAM, LSD-SLAM and ORB-SLAM on the KITTI dataset.

	e_F [m]			
	SOFT	LG-SLAM	ORB-SLAM	LSD-SLAM
KITTI00	3.36	1.18	1.3	**1.0**
KITTI02	5.52	3.12	5.7	**2.6**
KITTI05	1.54	**0.59**	0.8	1.5
KITTI06	0.96	**0.49**	0.8	1.3
KITTI07	0.4	**0.32**	0.5	0.5
KITTI09	2.42	**1.26**	3.2	5.6

Table 2. Results of LG-SLAM, LSD-SLAM and ORB-SLAM on the EuRoC dataset. MH stands for datasets recorded in machine hall, while V stands for dataset recorded in the Vicon room. E, M and D depict easy, medium and difficult sequences respectively.

	e_F [cm]			
	SOFT	LG-SLAM	ORB-SLAM	LSD-SLAM
MH_01_E	17.2	**3.6**	4.0	-
MH_02_E	7.8	5.0	**4.3**	-
MH_03_M	16.8	3.9	**3.5**	-
MH_04_D	32.8	7.5	**7.1**	-
MH_05_D	24.4	6.0	**5.3**	-
V1_01_E	10.8	**4.8**	8.7	6.6
V1_02_M	14.0	**4.8**	6.4	7.4
V1_03_D	32.7	**4.7**	7.2	8.9
V2_01_E	16.2	7.0	**6.1**	-
V2_02_M	22.4	8.2	**5.6**	-

for the pure SOFT odometry, which is the only front-end used in this scenario, since both LSD-SLAM and ORB-SLAM are stereo visual SLAM solutions. Since the code of ORB-SLAM is available as open source, we tried using it to get results for the last sequence. However, results for all sequences were slightly worse than the ones stated in the paper, which is why we do not report them here.

Given the results on the KITTI dataset in Table 1, we can see that LG-SLAM achieved the best results on all the tracks except in two cases. In particular, LSD-SLAM shows the best performance on sequences KITTI00 and KITTI02. It can also be seen that LG-SLAM significantly improved SOFT results in all the tracks. Figure 4 shows LG-SLAM, SOFT and ground truth trajectories for the sequence KITTI00. When we analyze the EuRoC dataset results shown in Table 2, we can notice that LG-SLAM outperformed both solutions in the sequences taken in the Vicon 1 room, while ORB-SLAM was better in 4 sequences from the Machine Hall and 2 sequences from Vicon room 2. As in the case of the KITTI dataset LG-SLAM again significantly improved the accuracy with respect to the SOFT odometry.

Additionally, we have compared our LG-SLAM online on the KITTI dataset, using the built-in evaluation protocol. For this purpose we tested LG-SLAM on the remaining 10 sequences and in Table 3 we can see the overall result

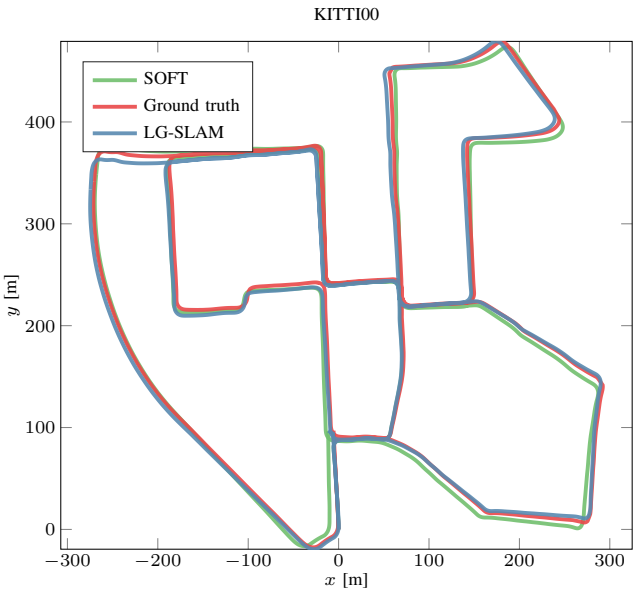

Figure 4. LG-SLAM results on the KITTI sequence KITTI00

Table 3. KITTI rankings of the state-of-the-art stereo vision SLAM systems at the time of writing.

Method	Transl.	Rot. [°/m]	Sensors
LG-SLAM	0.82 %	0.0020	stereo cameras
ORB-SLAM2	1.15 %	0.0027	stereo cameras
S-PTAM	1.19 %	0.0025	stereo cameras
S-LSD-SLAM	1.20 %	0.0033	stereo cameras

achieved by LG-SLAM together with other three best ranked SLAM solutions (two of them being ORB-SLAM and LSD-SLAM). The complete results with details are also available online[2], and, at the time of writing, the proposed approach ranked second among the stereo vision approaches and first among the tested SLAM solutions.

7.2 Experimental comparison of LG-SLAM and g^2o

Even though both LSD-SLAM and ORB-SLAM use g^{2}o as the back-end, their front-ends are different. This is why, in order to conduct a fair comparison between LG-SLAM and g^{2}o, we have used clean g^{2}o and LG-SLAM back-ends and coupled them with the SOFT and 3D-NDT front-ends. We have also ensured that both back-ends have exactly the same number of states, that states are added precisely after the same stereo pair or point cloud was processed, and that the same uncertainty model is used. To see how g^{2}o relates to other solvers in a pose graph initialization and rotation estimation study, the reader is referred to the paper of Carlone et al. (2015).

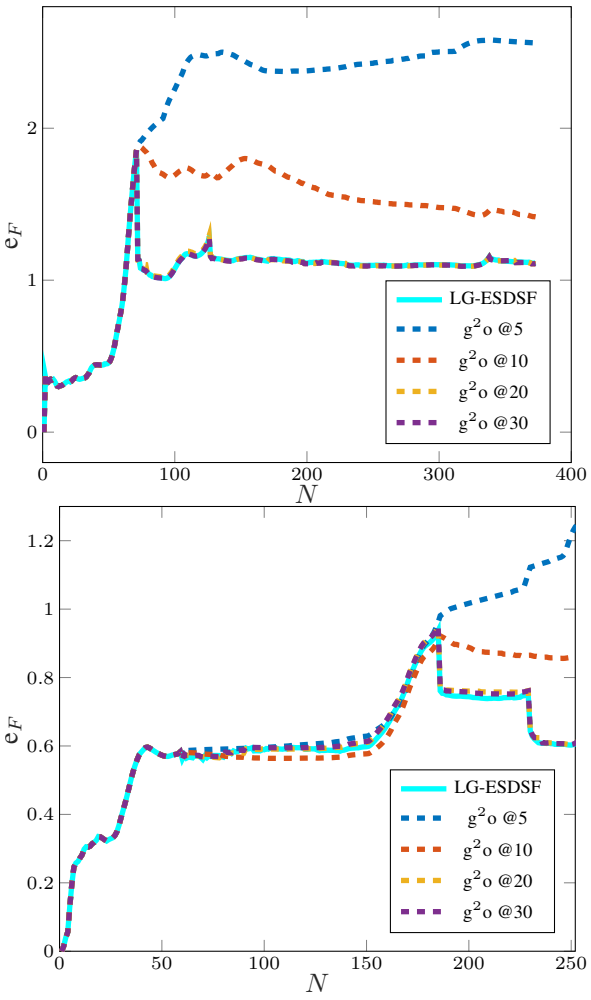

Figure 5. Comparison of e_F through time for different g^{2}o maximum iteration values and e_F of LG-ESDSF. N is the number of states. Upper figure shows e_F for sequence KITTI00 and the bottom figure for sequence KITTI05.

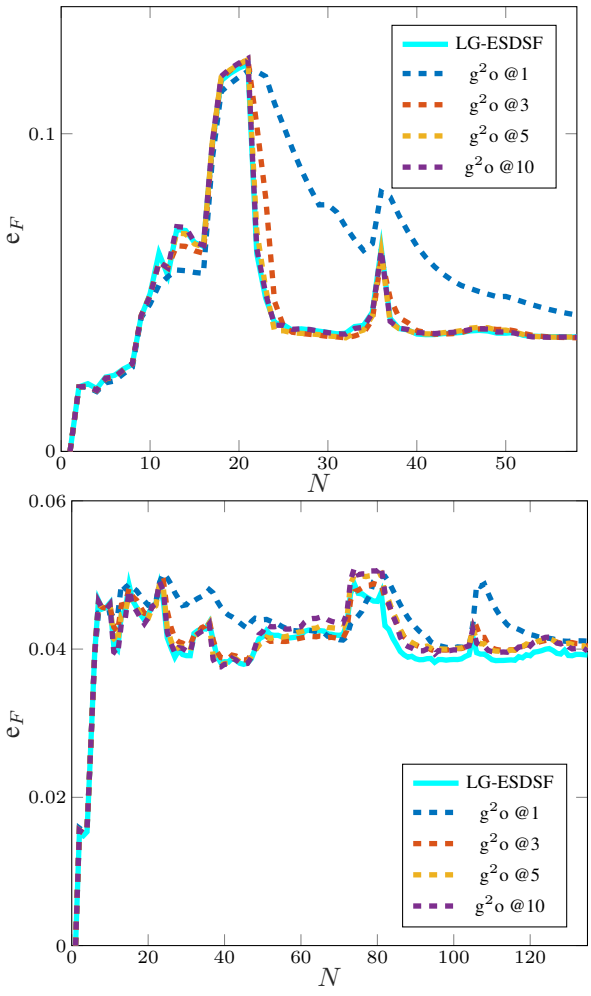

Figure 6. Comparison of e_F through time for different g^{2}o maximum iteration values and e_F of LG-ESDSF. N is the number of states. Upper figure shows e_F for EuRoC track MH_01_E and bottom figure for EuRoC track MH_01_M.

7.2.1 Number of iterations. Since g^{2}o converges to the solution in an iterative fashion, the parameter which limits the number of iterations drastically impacts its performance. The overall SLAM performance is affected not only by accuracy of the used back-end, but also by its computation speed, and hence the maximum number of iterations during optimization has to be limited. In order to determine the number of iterations which balances between speed and accuracy, we have evaluated g^{2}o behaviour on two different tracks of each dataset limiting the number of iterations to several different values. Finally, we have picked the limit for EuRoC and KITTI datasets, such that an increase in the number of iterations only insignificantly improves accuracy.

Figures 5 and 6 show e_F consisting of states appearing in every 10-th step of the trajectory and states appearing whenever updates were performed. Both algorithms were relying on the same front-end solutions. The maximum number of iterations on KITTI sequences was limited to 5, 10, 20 and 30, while on EuRoC sequences it was limited to 1, 3, 5 and 10 since it saturated more quickly.

In the sequel, for testing purposes, on KITTI dataset we set the maximum number of iterations to 20, since the accuracy was saturated at this point. Following the similar reasoning and given Fig. 6, the maximum number of iterations for EuRoC dataset was set to 5. From Figs. 5 and 6 we can see that on the tested tracks, the error of LG-ESDSF is similar to that of g^{2}o with the chosen maximum number of iterations.

The results of both back-end algorithms relying on SOFT and 3D-NDT front-ends on the KITTI dataset, are listed in

Table 4. Comparison of g^{2}o and LG-SLAM on the KITTI dataset with the SOFT front-end showing translational, rotational and total error.

	e_{trans} [m]	e_{rot} [deg]	e_F
	LG-SLAM / g^{2}o		
KITTI00	4.41 / **4.40**	1.25 / 1.25	1.10 / **1.09**
KITTI02	13.02 / **12.90**	1.21 / **1.16**	**3.44** / 3.45
KITTI05	1.45 / 1.45	0.55 / 0.55	**0.56** / 0.57
KITTI06	1.26 / 1.26	0.81 / 0.81	**0.62** / 0.63
KITTI07	0.93 / 0.93	0.52 / 0.52	0.32 / 0.32
KITTI09	2.72 / 2.72	0.77 / 0.77	1.24 / 1.24

Table 5. Comparison of g^{2}o and LG-SLAM on the KITTI dataset with the 3D-NDT front-end showing translational, rotational and total error.

	e_{trans} [m]	e_{rot} [deg]	e_F
	LG-SLAM / g^{2}o		
KITTI00	8.90 / **8.83**	**3.62** / 3.74	**2.28** / 2.33
KITTI02	**98.27** / 130.97	**19.57** / 24.75	**46.65** / 49.71
KITTI05	2.87 / **2.85**	1.92 / **1.87**	1.34 / 1.34
KITTI06	3.27 / **3.16**	2.01 / **1.92**	1.95 / **1.87**
KITTI07	1.22 / **1.21**	**0.97** / 0.99	0.64 / 0.64
KITTI09	14.81 / **14.78**	4.05 / 4.05	3.34 / 3.34

Table 6. Comparison of g^{2}o and LG-SLAM on the EuRoC dataset.

	e_{trans} [m]	e_{rot} [deg]	e_F
	LG-SLAM / g^{2}o		
MH_01_E	**0.16** / 0.17	**2.10** / 2.59	0.04 / 0.04
MH_02_E	0.11 / 0.11	**1.05** / 1.18	0.05 / 0.05
MH_03_M	0.13 / 0.13	**2.10** / 2.18	0.04 / 0.04
MH_04_D	0.29 / **0.28**	**0.95** / 1.08	**0.07** / 0.09
MH_05_D	**0.16** / 0.17	**0.98** / 1.02	**0.06** / 0.08
V1_01_E	0.25 / **0.24**	**2.32** / 2.35	0.05 / 0.05
V1_02_M	0.09 / 0.09	**0.88** / 0.94	0.05 / 0.05
V1_03_D	**0.12** / 0.13	**1.51** / 1.79	0.05 / 0.05
V2_01_E	**0.12** / 0.15	**1.64** / 2.53	**0.07** / 0.09
V2_02_M	0.08 / 0.08	1.76 / **1.70**	0.08 / **0.07**

Tables 4 and 5. By examining results produced by back-ends, we can conclude that in most cases the accuracy is very similar, with minor differences. The exception is the sequence KITTI02 when relying on 3D-NDT front-end, since it accumulated high rotational error in the beginning and the loop was never closed. In this case, none of the back-end approaches were able to produce accurate result.

Since EuRoC dataset contains only stereo images, in this case we relied on the SOFT front-end only. Results for all the 10 sequences are provided in Table 6. Again, both back-ends achieve similar performance in terms of accuracy, with LG-SLAM performing slightly better in terms of rotation error e_{rot}. However, even the maximum difference in e_{rot} is smaller than 1°, and in most sequences it is even smaller than 0.1°.

Table 7. Minimum, maximum and mean computation times of the LG-SLAM update step and g^{2}o optimization on the KITTI dataset with the SOFT front-end.

	t_{min} [ms]	t_{max} [ms]	t_{mean} [ms]
	LG-SLAM / g^{2}o		
KITTI00	10.6 / 1.5	37.1 / 996.7	27.2 / 461.2
KITTI02	29.3 / 1.7	39.1 / 998.9	34.9 / 349.6
KITTI05	8.4 / 35.7	21.9 / 540.3	14.4 / 250.0
KITTI06	7.4 / 32.4	12.3 / 269.5	9.6 / 165.1
KITTI07	5.8 / 158.1	7.2 / 195.5	6.4 / 178.6
KITTI09	15.2 / 439.9	16.1 / 448.4	15.7 / 443.2

Table 8. Minimum, maximum and mean computation times of the LG-SLAM update step and g^{2}o optimization on KITTI dataset with the 3D-NDT front-end.

	t_{min} [ms]	t_{max} [ms]	t_{mean} [ms]
	LG-SLAM / g^{2}o		
KITTI00	10.6 / 1.8	42.1 / 999.8	27.0 / 510.1
KITTI02	29.3 / 181.6	40.4 / 972.2	35.6 / 829.8
KITTI05	8.4 / 49.6	22.4 / 554.7	14.3 / 278.0
KITTI06	7.1 / 33.0	12.8 / 297.0	9.5 / 157.2
KITTI07	5.8 / 56.4	7.2 / 175.1	6.3 / 124.3
KITTI09	14.6 / 406.0	16.5 / 438.5	15.4 / 420.7

7.2.2 Computation runtime. The runtimes of both back-end algorithms relying on SOFT and 3D-NDT front-ends on KITTI dataset, are given in Tables 7 and 8. The maximum t_{max}, minimum t_{min} and mean t_{mean} computation times for all 6 sequences are presented. The number of iterations for g^{2}o was set to the valued determined in Section 7.2.1. It can be seen that all the computation times of the LG-SLAM update steps are significantly smaller than those of the g^{2}o optimization. Furthermore, since update of LG-SLAM is only dependent on the number of loop closings and number of states in the trajectory, we can see that computation times for LG-SLAM are similar regardless of the used front-end. On the other hand, g^{2}o optimization runtimes depend on the initial condition. It can also be noted that the range of runtimes for g^{2}o is much wider, which is due to faster optimization if loop closings appear frequently, hence in KITTI07 and KITTI09 minimum and maximum optimization runtimes are similar, since there were very few loop closings. Table 9 shows computation times of the update step for LG-SLAM and optimization step for g^{2}o on EuRoC dataset. Again, same conclusions can be drawn as in the case of the KITTI dataset.

7.2.3 Robustness to bias. Additionally, we compared the behaviour of the two back-ends during specific conditions which may occur and are important performance indicators for a SLAM system. These conditions include: (i) existence of bias in the odometry measurements, (ii)

Table 9. Minimum, maximum and mean computation times of the LG-SLAM update step and g^{2}o optimization on the EuRoC dataset.

	$t_{\min}$ [ms]	$t_{\max}$ [ms]	t_{mean} [ms]
	LG-SLAM / g^{2}o		
MH_01_E	0.3 / 1.3	3.4 / 27.5	2.1 / 17.6
MH_02_E	0.2 / 0.5	3.6 / 26.5	2.2 / 16.3
MH_03_M	0.2 / 0.8	8.2 / 53.7	3.6 / 32.0
MH_04_D	0.2 / 1.1	4.9 / 35.3	2.8 / 21.3
MH_05_D	0.4 / 2.1	5.4 / 37.2	2.9 / 23.0
V1_01_E	0.3 / 1.7	3.7 / 28.6	2.2 / 15.8
V1_02_M	0.2 / 1.0	4.5 / 39.5	2.1 / 18.3
V1_03_D	0.4 / 2.1	3.5 / 30.9	2.2 / 18.8
V2_01_E	0.6 / 3.6	1.5 / 12.1	1.0 / 7.9
V2_02_M	0.2 / 1.0	4.0 / 29.6	1.8 / 15.5

large error introduced in several relative pose measurements after loop closing detection, (iii) large error introduced in several odometry measurements and (iv) perform only one update/optimization after the end of experiment. For testing purposes we used the KITTI00 track.

Bias in the odometry measurements was introduced by adding a fixed value to every odometry measurement produced by the SOFT front-end, primarily to the yaw angle. Figure 7 shows comparison between the ground truth and odometry trajectories with and without bias. As can be seen, introducing bias has severely degraded the odometry accuracy. However, both SLAM back-ends have successfully corrected the introduced error as can be seen in Fig. 8, which shows continuous plot of error e_F. The difference in e_F between the two approaches is very small, i.e., g^{2}o finished with $e_F = 5.50$ and LG-ESDSF finished with $e_F = 5.64$. Moreover, error difference at every point is very small and we can conclude that both SLAM back-ends can accurately estimate trajectory even in the presence of significant bias.

7.2.4 Robustness to outliers. In contrast to bias, outliers appearing as large errors that occur only several times during the experiment, cannot be well modeled with appropriate covariances. This makes outliers more challenging for a SLAM system to recover from.

In the first experiment we introduced outliers in relative pose measurements at every 20-th loop closing. This was performed by perturbing the accurate relative pose estimate with an $\mathsf{SE}(3)$ element Δ_e. The error was set to 30° in rotation and 10 m in translation. Figure 9 shows continuous plots of e_F for g^{2}o and LG-ESDSF. The outlier occurrence can be recognized by high jumps in e_F. As can be seen from the results, although the final e_F of both trajectories is close ($e_F = 5.12$ for g^{2}o and $e_F = 5.28$ for LG-ESDSF), by examining the entire graph we can see that g^{2}o achieved lower error, especially between states 250 and 350. The

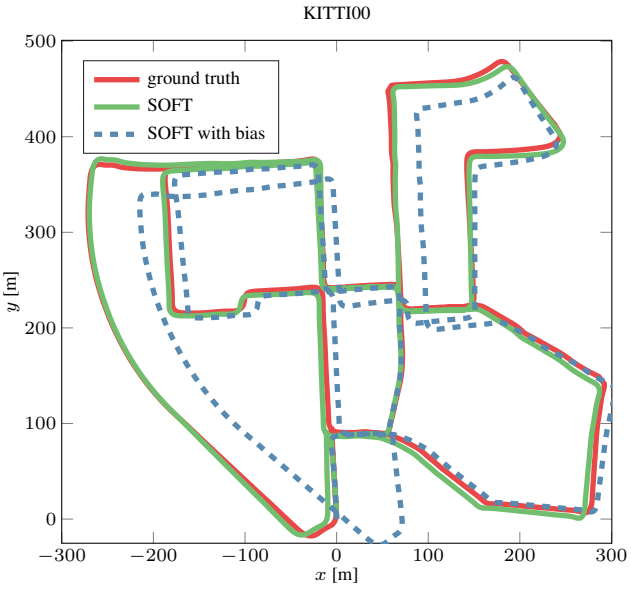

Figure 7. Comparison between ground truth trajectory and SOFT trajectories with and without introduced bias.

main reason for this lies in the g^{2}o relinearization ability occuring at every iteration.

The second experiment is performed so that the outliers in the odometry were introduced in the same way as in the case of relative pose measurements of the loop closings, but with lower absolute values on angles and location displacements of Δ_e. The maximum orientation change per angle was set to 20° and maximum position displacement was set to 5 m. The outliers were introduced every 800 steps. The continuous plot of e_F with odometry outliers is shown in Fig. 10, while the final errors are $e_F = 4.28$ for g^{2}o, and $e_F = 5.63$ for LG-ESDSF. Again, as is the case of outliers in relative pose measurements, odometry outliers were better corrected with g^{2}o.

7.2.5 LG-ESDSF as a general optimization tool. In order to test the ability of LG-ESDSF to perform as a general solution for optimization problems formulated via graph structure in a similar fashion as g^{2}o, we compared the two back-ends in the case when only one update is performed at the end of the sequence. In this experiment g^{2}o was set so that all edges resulting from odometry and loop closing detections are added as they occur, however, only one optimization is called at the end of the sequence. LG-ESDSF was set so that equation (40) is slightly modified. In particular, Jacobian matrix $\mathcal{H}$ is no longer of dimensions $6 \times N$, but becomes matrix with dimensions $6n_u \times N$, where n_u is total number of loop closing detections. Covariance matrix R is no longer of dimensions 6×6, but a block-diagonal matrix $6n_u \times 6n_u$, and innovation vector is no longer 6×1, but $6n_u \times 1$. Each $6 \times N$ row in $\mathcal{H}$

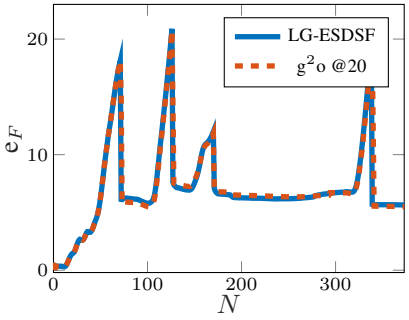

Figure 8. Continuous error plot e_F of g^{2}o and LG-ESDSF trajectories when bias was added to the SOFT odometry measurements.

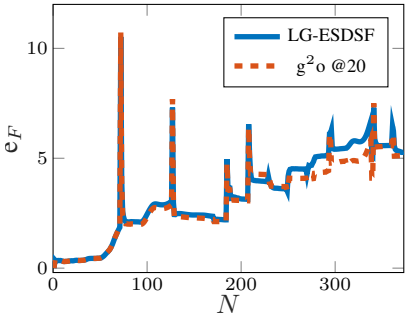

Figure 9. Continuous error plot e_F of g^{2}o and LG-ESDSF trajectories when outliers were introduced in the relative pose measurements.

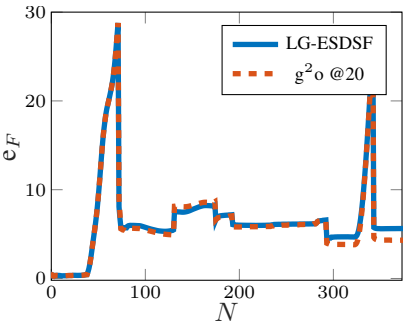

Figure 10. Continuous error plot e_F of g^{2}o and LG-ESDSF trajectories when outliers were introduced in the odometry measurements.

Table 10. Comparison of g^{2}o and LG-SLAM on KITTI00 with only one update/optimization performed at the end.

	e_{trans} [m]	e_{rot} [deg]	e_F	t_u [ms]
LG-ESDSF	4.41	1.25	1.10	55.2
g^{2}o 10	9.70	2.00	3.04	483
g^{2}o 20	4.40	1.25	1.10	978
g^{2}o 30	4.40	1.25	1.09	1078

is added after every loop closing and is calculated in the same way as when update is performed after each loop closing detection. Each block of R is added with every new row in matrix $\mathcal{H}$ and represents uncertainty of the relative pose measurement for the particular loop closing detection. Innovation vector z is also extended after every loop closing detection with 6×1 vector representing the innovation resulting from the occurred loop closing detection. Once information from all the updates is stored in matrices $\mathcal{H}$, R and z, and the sequence is completed, update step is performed using (40) and (42)–(44).

Table 10 shows errors e_F, e_{tran} and e_{rot} of the final trajectory after single update/optimization was performed at the end of KITTI00 sequence. Additionally, the runtimes t_u representing duration required for the update/optimization to be completed are also presented. From the results we can see that accuracy of both LG-ESDSF and g^{2}o has remained the same as in the case when multiple updates appeared sequentially over time. However, when maximum number of iterations during optimization was limited to 10, results were significantly less accurate for g^{2}o, while with 30 iterations there was no improvement in the accuracy of the final trajectory. However, LG-ESDSF remained significantly faster than g^{2}o, while retaining same level of accuracy. Figure 11 shows KITTI00 ground truth trajectory, as well as the estimated trajectories obtained by g^{2}o and LG-ESDSF.

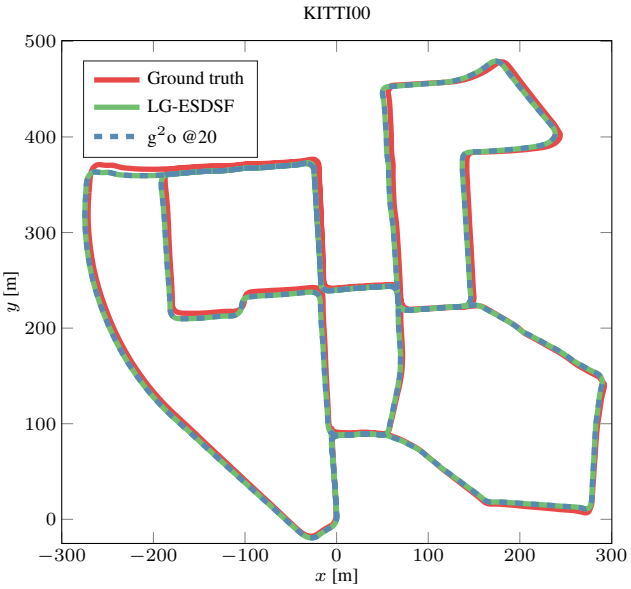

Figure 11. Comparison between ground-truth, g^{2}o and LG-ESDSF trajectories after only one update/optimization step performed at the end.

7.2.6 Discussion. Besides taking the advantage of a Lie group state representation, g^{2}o uses the Gauss-Newton or Levenberg-Marquardt algorithm to solve SLAM represented as a least squares problem in an iterative fashion, relinearizing after each iteration of the algorithm and minimizing a weighted error function. It is a general framework for solving nonlinear least squares problems via optimization that can be represented as a graph, thus being a highly general and extensible method, not only applicable to SLAM problems. LG-ESDSF, on the other hand, operates in only a single linearization point used for obtaining the measurement Jacobian $\mathcal{H}$, and instead of gradually reaching optimization solution through several iterations, it calculates the updated estimate based on a

single innovation term (40). Nevertheless, a similarity between the compared back-ends exists, and stems from the fact that LG-ESDSF also tackles the SLAM problem as a pose graph and keeps the history of the whole trajectory, but in the form of the information vector within a filtering framework, which is different from most EKF based SLAMs that keep the current robot pose and the pose of the landmarks. Hence, when ESDSF computes the update step, it does so over the entire trajectory, thus can be seen as a form of trajectory smoothing (Dellaert and Kaess (2006)).

Furthermore, compared to approaches treating orientation parametrization as an Euclidean vector space with a pertaining covariance matrix, which can be reasonable for small rotations when the non-Euclidean nature is not as pronounced, by working on Lie groups and manipulating covariances correspondingly, the uncertainty interrelations can be better captured, which is especially important for filtering frameworks, since the goal of the Kalman gain is to ensure optimal update by considering these exact covariance interrelations. For example, direct filtering over Euler angle or unit quaternion parametrizations ignores the non-Euclidean nature which is in play between each of the parametrization components.

Moreover, it also holds that a filtering based SLAM can rely on filtering relinearization approaches, such as iterative EKF (Lie group solution was presented in Bourmaud et al. (2016)), and indeed in Bell and Cathey (1993) and Bell (1994) it was shown that the iterative EKF update is an application of the Gauss-Newton method for approximating a maximum likelihood estimate and that iterated Kalman smoother is a Gauss-Newton method for maximizing the likelihood function. Nevertheless, in the present paper we focus on the standard single step filtering.

In the experiments, with LG-ESDSF we have demonstrated that accuracy with respect to g^{2}o remained comparable in most scenarios, exempting outliers where g^{2}o showed higher accuracy, while runtime was significantly lower. Although in general g^{2}o would most likely as a batch optimization approach achieve better accuracy given an unlimited number of iterations, in practical SLAM applications this number has to be limited in order to assure real-time operation. Hence, after carefully selecting the limits on the number of iterations during optimization, LG-ESDSF was shown to perform the update step 10-25 times faster on the KITTI dataset and 7-8 times faster on the EuRoC dataset with respect to g^{2}o. The reason behind KITTI and EuRoC timing differences lies in the fact that the EuRoC dataset contains shorter trajectories with more often occurring loop closings. Moreover, the runtime of LG-ESDSF can be predicted and accounted for.

In conclusion, we can assert that by negotiating uncertainties and employing filtering equations on Lie groups within the ESDSF framework, thus respecting the state space geometry, LG-SLAM managed to achieve state-of-the-art performance based on a veteran filtering SLAM approach.

7.3 Evaluation of the long-term LG-SLAM performance

In order to test Algorithm 2 we again used the KITTI dataset, particularly sequences KITTI00 and KITTI05. We have chosen these sequences because they are among the longest ones and they have significant trajectory parts that overlap and a loop closing near the beginning. For the testing purposes we have simulated two consecutive runs of each trajectory and used SOFT for odometry. Although this does not completely simulate a real-world experiment, since images of the second run would not be identical, they would be similar enough. Since our goal in this scenario was not to test the accuracy of the front-end, we think that this approach is adequate to test the long-term LG-SLAM approach. Another advantage of such a simulation is that, since images are identical, trajectory augmentation with almost every new state in the second run results in a loop closing, which is precisely the scenario we wanted to test. The benefit is also that we had accurate ground-truth for both experiments.

To test the behaviour of our algorithm we first ran the simulation on both tracks with two runs without using the algorithm in order to get a reference. Afterwards, we used the topological distance to reject unnecessary loop closings and performed the marginalization of all states that were selected as close enough between two consecutive loop closings, but without including the state participating in the loop closing. This means that no sparsification was required. Finally, we performed testing of all the components by also including the state in which the loop closing was detected. We have evaluated accuracy of final tracks using e_F, counted the number of states n in the trajectory at the end, recorded total number of updates performed n_u, measured the total time required for all trajectory updates t_u, and counted states added in the second run n_{new}.

Table 11 shows the results for KITTI00. Label NO_MARG stands for results acquired without using the Algorithm 2. Label NO_LOOP_MARG stands for results when the state in which loop closing occurred was not marginalized and label ALL stands for results when all the steps of the algorithm were used. We tested the algorithm performance for two drastically different values of the minimum topological distance ${}^{T}d_{\min}$, one was set to 80 m and the other to 6 m. We can see that when the algorithm was not used, error e_F for both runs seen in Table 11 remained the same as the error of the single run (confer e_F in Table 4). As expected, when rejecting updates and marginalizing blocks the error increases, while in the case

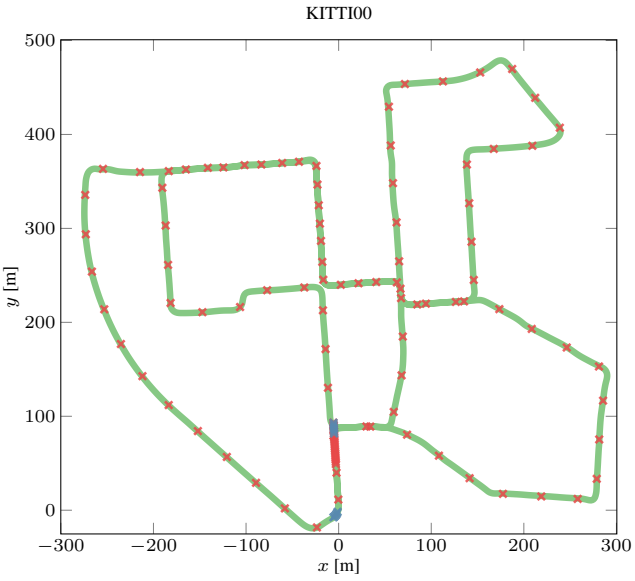

Figure 12. New states added in the second run of KITTI00 with ${}^{T}d_{min} = 80$ m. Green line represents trajectory in the first run, red $\times$ represent states added in the second run for the NO_LOOP_MARG case, and blue $\times$ represent states added for the ALL case. We can notice a significant reduction of added redundant states for the ALL case.

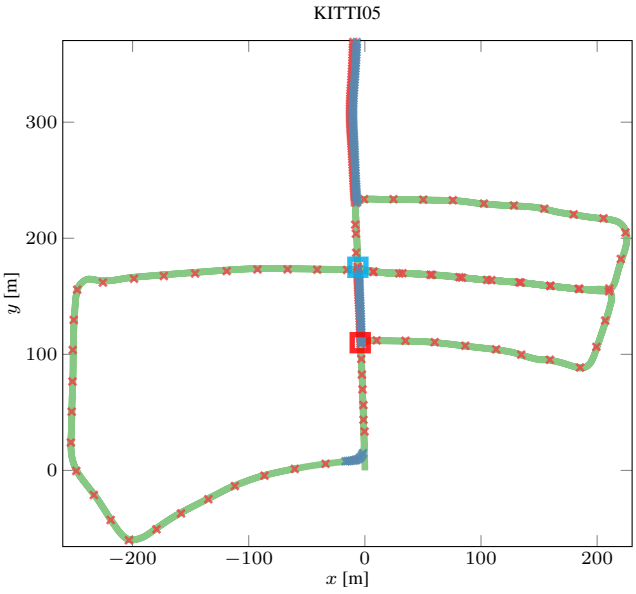

Figure 13. New states added in the second run of KITTI05 with ${}^{T}d_{min} = 50m$. Green line represents trajectory in the first run, red $\times$ represent states added in the second run for the NO_LOOP_MARG case, and blue $\times$ represent states added for the ALL case. We can notice a significant reduction of added redundant states for the ALL case. Red rectangle marks the area where the vehicle went right and missed the loop closing in the area marked with cyan rectangle during the first run, while in the second run it continued straight at the red rectangle and closed the loop at the cyan rectangle.

of using sparsification, the error increase is even further noticeable. However, for all the tests the error increase remained small. When using the smaller ${}^{T}d_{\min}$, changes in the error were almost insignificant, since, in that case, fewer updates were rejected. This can be seen from the parameter n_u, which also indicates the correct behaviour of the loop closing rejection part. From Fig. 12 we can see that, when states in which update occurred were not marginalized, they were periodically added to the trajectory after each accepted update (red crosses in the figure). However, updates were performed only after ${}^{T}d_{\min}$ was exceeded. When states in which loop closing occurred were marginalized, only few states were added in the second run (blue crosses in the figure). This can also be confirmed by consulting the parameter n_{new}.

The most interesting parameter is probably t_u. As we can see when ${}^{T}d_{\min}$ is large, time required to complete all updates is practically the same when sparsification is enabled or disabled. This is because very few updates were performed and the effect of more dense information matrix was nullified by the time required to do sparsification. Much better effect of sparsification can be seen when ${}^{T}d_{min}$ is small. In this case we can see that t_u is much smaller when using sparsification in comparison to the case when we do not. This confirms the expected effect of marginalizing the state in which loop closing was detected.

Results from the test conducted using KITTI05 are displayed in Table 12. Most of the conclusions drawn for the test using KITTI00 can also be made here. The difference is that in this case sparsification reduces accuracy even less. We can see that in this case e_F is even slightly smaller when using marginalization with small ${}^{T}d_{\min}$ than without performing any marginalization, which is due to rejecting some loop closings that had errors in their estimates. The e_F of two runs seen in Table 11 in this case is larger than in the case of only one run (confer e_F in Table 4). The reason for this is, because unlike in KITTI00, in KITTI05 two runs are different and one important loop closing is not detected until the end of the second run. This can be seen in Fig. 13. When robot arrived at the position marked with red rectangle in the first run, it turned right, while in the second run it went straight and closed the loop in the area marked with the cyan rectangle. From this figure we can also see that when robot entered a previously unexplored area (straight segments densely populated with crosses) the state vector was correctly augmented with new states.

From the results we can say that the proposed algorithm works correctly and drastically reduces the number of unnecessary states in the trajectory. However, we do

Table 11. Long-term performance evaluation on KITTI00

	e_F [m]	n	n_u	t_u [s]	n_{new}
NO_MARG	1.17	3865	2215	130.62	1949
$^{T}d_{min} = 80m$					
NO_LOOP_MARG	1.21	1758	106	2.77	109
ALL	1.27	1645	121	2.86	12
$^{T}d_{min} = 6m$					
NO_LOOP_MARG	1.18	2772	1140	47.42	986
ALL	1.19	1646	1183	27.81	18

Table 12. Long-term performance evaluation on KITTI05

	e_F [m]	n	n_u	t_u [s]	n_{new}
NO_MARG	1.04	2150	1163	35.7	1149
$^{T}d_{min} = 50m$					
NO_LOOP_MARG	1.29	1072	88	1.31	172
ALL	1.29	997	96	1.30	107
$^{T}d_{min} = 5m$					
NO_LOOP_MARG	1.0	1633	650	14.58	669
ALL	1.01	993	669	8.31	105

acknowledge that the process of choosing which states should be removed by checking the similarity of the pose has its drawbacks and cannot, for example, account for changes in the environment like moving objects and occlusions. In order to make this algorithm applicable in such conditions, we would have to use information from the SLAM front-end which is out of scope of the present paper and subject of future work. On the other hand, all other steps of the proposed algorithm are front-end agnostic.

8 Conclusion

The successful solution of the SLAM problem is one of the main prerequisites for the autonomy of every mobile robot. Development of SLAM back-end algorithms is usually divided into two main branches. One branch is focused on developing SLAM solutions based on filtering and the other is centred on using graph-optimization algorithms. In the last decade, solutions based on graph-optimization prevailed over the filtering approaches. However, with the introduction of new EKF filtering approaches that solve equations directly on Lie groups, a novel ground was set for filtering based SLAM which respects the geometry of the state space, thus achieving greater estimation accuracy of both the mean and the covariance.

In this paper we have proposed a novel filtering SLAM back-end solution based on the ESDSF derived on Lie groups, dubbed LG-ESDSF. The developed filter does not only retain all the good characteristics of the classic ESDSF implementation, the main being the exact sparsity of the information matrix, but also respects the state space geometry by negotiating uncertainties and employing filtering equations on Lie groups. In addition, we have developed a method which allows our LG-ESDSF back-end to work over long-term, while the robot is continuously moving through already explored environments. In order to test the LG-ESDSF back-end, we have coupled it with two different front-ends: one based on the stereo camera and one based on the 3D laser range sensor. We named our complete SLAM solution LG-SLAM. For testing we used two publicly available datasets. First dataset was the KITTI dataset which was recorded with a commercially available vehicle equipped with a 3D LIDAR and two pairs of stereo cameras. The second dataset was the EuRoC dataset recorded with an UAV equipped with a stereo camera. We compared results of LG-SLAM with g^{2}o when coupled with the same front-ends and under the same conditions. Results have showed that the proposed method achieves similar accuracy as g^{2}o, with significantly faster computation times of the update step in comparison to the g^{2}o optimization. We have also evaluated LG-SLAM using the KITTI online evaluation protocol achieving the second best result among all the stereo odometry solutions and the best results among all the tested SLAM algorithms. Finally, we tested LG-SLAM long-term performance using the KITTI dataset. The results validated the effectiveness of LG-SLAM in such conditions. In the end we can assert that although graph-optimization solutions still offer several advantages over the filtering solutions, like easier applicability to a wider range of problems, judging from the newly proposed method, we believe that filtering solutions deserve to be once again in the focus of the SLAM research.

9 Appendices

9.1 ESDSF equations for augmentation and marginalization during the prediction

The derivation of Euclidean ESDSF is given in Eustice et al. (2006), while here we provide only the final expressions.

9.1.1 Trajectory augmentation. If we denote estimated pose in time step n by X_n, and all other previous poses with M, i.e., $M = [X_1, \ldots, X_{n-1}]^T$, then we can partition the joint probability distribution of X_n and M as follows

$$\begin{aligned} p(X_n, M | z_{1:n}, u_{1:k}) &= \mathcal{N}\left[\begin{pmatrix} \mu_{X_n} \\ \mu_M \end{pmatrix}, \begin{pmatrix} \Sigma_{X_n} & \Sigma_{X_n M} \\ \Sigma_{M X_n} & \Sigma_{MM} \end{pmatrix}\right] \\ &= \mathcal{N}^{-1}\left[\begin{pmatrix} \eta_{X_n} \\ \eta_M \end{pmatrix}, \begin{pmatrix} \Lambda_{X_n X_n} & \Lambda_{X_n M} \\ \Lambda_{M X_n} & \Lambda_{MM} \end{pmatrix}\right]. \end{aligned} \tag{58}$$

where $z_{1:n}$ represents history of all measurements and $u_{1:k}$ represents history of all control inputs. Augmentation step yields

$$p(X_{n+1}, X_n, M | z_{1:n}, u_{1:k}) = \mathcal{N}^{-1}(\eta_{n+1}, \Lambda_{n+1}), \tag{59}$$

where

$$\eta_{n+1} = \begin{bmatrix} Q_n^{-1}(\mu_{X_{n+1}} - F_n\mu_{X_n}) \\ \eta_{X_n} - F_n^T Q^{-1}(\mu_{X_{n+1}} - \mu_{X_n}) \\ \eta_M \end{bmatrix} \quad (60)$$

and

$$\Lambda_{n+1} = \begin{bmatrix} Q_n^{-1} & -Q_n^{-1}F_n & 0 \\ -F_n^T Q_n^{-1} & \Lambda_{X_{n,n}} + F_n^T Q_n^{-1} F_n & \Lambda_{X_n M} \\ 0 & \Lambda_{MX_n} & \Lambda_{MM} \end{bmatrix}. \quad (61)$$

9.1.2 State marginalization. Marginalization of the state X_n equals marginalization of a Gaussian from a multivariate Gaussian distribution T_n

$$\begin{aligned} p(X_{n+1}, M|z_{1:n}, u_{1:k}) &= \int p(X_{n+1}, X_n, M|z_{1:n}, u_{1:k}) dX_n \\ &= \mathcal{N}(\bar{\mu}_n, \bar{\Sigma}_n) = \mathcal{N}^{-1}(\bar{\eta}_n, \bar{\Lambda}_n) \end{aligned} \quad (62)$$

where

$$\begin{aligned} \bar{\eta}_n &= \begin{bmatrix} Q_n^{-1}(\mu_{X_{n+1}} - F_n\mu_{X_n}) \\ \eta_M \end{bmatrix} \\ &\quad - \begin{bmatrix} -Q_n^{-1}F_n \\ \Lambda_{MX_n} \end{bmatrix} \alpha_n^{-1}(\eta_{X_n} - F_n^T Q_n^{-1}(\mu_{X_{n+1}} - F_n\mu_{X_n}) \\ &= \begin{bmatrix} Q_n^{-1}F_n\alpha_n^{-1}\eta_{X_n} + \beta_n(\mu_{X_{n+1}} - F_n\mu_{X_n}) \\ \eta_M - \Lambda_{MX_n}(\eta_{X_n} - F_n^T Q_n^{-1}(\mu_{X_{n+1}} - F_n\mu_{X_n})) \end{bmatrix} \end{aligned} \quad (63)$$

$$\begin{aligned} \bar{\Lambda}_n &= \begin{bmatrix} Q_n^{-1} & 0 \\ 0 & \Lambda_{mm} \end{bmatrix} - \begin{bmatrix} -Q_n^{-1}F_n \\ \Lambda_{MX_n} \end{bmatrix} \alpha_n^{-1} \begin{bmatrix} -F_n^T Q_n^{-1} & \Lambda_{X_n M} \end{bmatrix} \\ &= \begin{bmatrix} \beta_n & Q_n^{-1}F_n\alpha_n^{-1}\Lambda_{X_n M} \\ \Lambda_{MX_n}\alpha_n^{-1}F_n^T Q_n^{-1} & \Lambda_{MM} - \Lambda_{MX_n}\alpha_n^{-1}\Lambda_{X_n M} \end{bmatrix}, \end{aligned} \quad (64)$$

and

$$\begin{aligned} \alpha_n &= \Lambda_{X_{n,n}} + F_n^T Q_n^{-1} F_n \\ \beta_n &= Q_n^{-1} - Q_n^{-1}F_n(\Lambda_{X_{n,n}} + F_n^T Q_n^{-1}F_n)^{-1}F_n^T Q_n^{-1} \\ &= (Q_n + F_n\Lambda_{X_{n,n}}^{-1}F_n^T)^{-1}. \end{aligned}$$

9.2 Special Euclidean group $\mathsf{SE}(3)$

The group $\mathsf{SE}(3)$ describes a 6 DoF rigid body pose and is formed as a semi-direct product of the Euclidean space vector $\mathbb{R}^3$ and the special orthogonal group $\mathsf{SO}(3)^3$, corresponding to translational and rotational parts, respectively. This group is defined as

$$\mathsf{SE}(3) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \subset \mathbb{R}^{4\times 4} \,|\, \{R, t\} \in \mathsf{SO}(3) \times \mathbb{R}^3 \right\}.$$

A Euclidean space vector representing the pose of a rigid body consisting of position $t = [t_1\, t_2\, t_3]$ and orientation $\phi = [\phi_1\, \phi_2\, \phi_3]$ vectors is obtained by concatenating the two, $x = [t\ \ \phi]^{\mathrm{T}}$. Mapping of the pertaining Euclidean space to Lie algebra, i.e., $(\cdot)^{\wedge}_{\mathsf{SE}(3)} : \mathbb{R}^6 \to \mathfrak{se}(3)$, is then constructed as

$$x^{\wedge}_{\mathsf{SE}(3)} = \begin{bmatrix} \phi^{\wedge}_{\mathsf{SO}(3)} & t \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(3), \quad (65)$$

$$\phi^{\wedge}_{\mathsf{SO}(3)} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathfrak{so}(3), \quad (66)$$

while its inverse, $(\cdot)^{\vee}_{\mathsf{SE}(3)} : \mathfrak{se}(3) \to \mathbb{R}^6$, follows trivially from (65) and (66). The exponential, performing mapping $\exp_{\mathsf{SE}(3)} : \mathfrak{se}(3) \to \mathsf{SE}(3)$, is determined as follows:

$$\exp^{\wedge}_{\mathsf{SE}(3)}(x) = \begin{bmatrix} C & Lt \\ \mathbf{0} & 1 \end{bmatrix} \quad (67)$$

$$\begin{aligned} C &= \exp^{\wedge}_{\mathsf{SO}(3)}(\phi) \\ &= \cos(|\phi|)I + (1 - \cos(|\phi|))\frac{\phi\phi^{\mathrm{T}}}{|\phi|^2} + \sin(|\phi|)\frac{\phi^{\wedge}_{\mathsf{SO}(3)}}{|\phi|} \\ L &= \frac{\sin(|\phi|)}{|\phi|}I + (1 - \frac{\sin(|\phi|)}{|\phi|})\frac{\phi\phi^{\mathrm{T}}}{|\phi|^2} + \frac{1-\cos(|\phi|)}{|\phi|^2}\phi^{\wedge}_{\mathsf{SO}(3)}. \end{aligned}$$

The logarithm, performing the mapping $\log_{\mathsf{SE}(3)} : \mathsf{SE}(3) \to \mathfrak{se}(3)$, is calculated by deconstructing X, and determining ϕ by using

$$\begin{aligned} \log_{\mathsf{SO}(3)}(X) &= \begin{cases} \dfrac{\theta}{2\sin(\theta)}(X - X^{\mathrm{T}}) & \text{if} \quad \theta \neq 0 \\ 0 \quad \text{if} \quad \theta = 0 \end{cases}, \\ &\text{s.t.} \quad 1 + 2\cos(\theta) = \mathrm{Tr(X)}. \end{aligned} \quad (68)$$

Then, from (67), we can determine t. In order to determine the adjoints for $\mathsf{SE}(3)$, we need to deconstruct the state $X \in \mathsf{SE}(3)$ and the vector $x \in \mathbb{R}^6$. Firstly, we extract the rotation part C and the translation part t from X, and secondly, we split the translation part t and the orientation part ϕ from x. Then, the adjoints $\mathrm{Ad}_{\mathsf{SE}(3)}$ and $\mathrm{ad}_{\mathsf{SE}(3)}$ are

$$\mathrm{Ad}_{\mathsf{SE}(3)}(X) = \begin{bmatrix} C & tC \\ \mathbf{0} & C \end{bmatrix}, \ \mathrm{ad}_{\mathsf{SE}(3)}(x) = \begin{bmatrix} \phi^{\wedge}_{\mathsf{SO}(3)} & t^{\wedge}_{\mathsf{SO}(3)} \\ 0 & \phi^{\wedge}_{\mathsf{SO}(3)} \end{bmatrix}$$

9.3 Derivation of the Lie measurement Jacobian $\mathcal{H}$

When the loop closing occurs between the states X_i and X_j we need to evaluate

$$\mathcal{H}_{n+1} = \frac{\partial}{\partial\epsilon}\left[\log^{\vee}_{\mathsf{G}}\left(h(X_i, X_j)^{-1}h\big((X_i, X_j)\exp^{\wedge}_{\mathsf{G}}(\epsilon)\big)\right)\right]\Big|_{\epsilon=0}$$

in order to perform update using (40). First we evaluate

$$\begin{aligned} h\big((X_i, X_j)\exp^{\wedge}_{\mathsf{G}}(\epsilon)\big) &= (X_j \exp^{\wedge}_{\mathsf{G}}(\epsilon_j))^{-1} X_i \exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\ &= \exp^{\wedge}_{\mathsf{G}}(-\epsilon_j) X_j^{-1} X_i \exp^{\wedge}_{\mathsf{G}}(\epsilon_i). \end{aligned}$$

Then, we evaluate

$$\begin{aligned}
&h(X_i, X_j)^{-1} h\big((X_i, X_j)\exp^{\wedge}_{\mathsf{G}}(\epsilon)\big) = \\
&= (X_j^{-1}X_i)^{-1}\exp^{\wedge}_{\mathsf{G}}(-\epsilon_j)X_j^{-1}X_i\exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\
&= X_i^{-1}X_j\exp^{\wedge}_{\mathsf{G}}(-\epsilon_j)X_j^{-1}X_i\exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\
&= X_i^{-1}\exp^{\wedge}_{\mathsf{G}}(\mathrm{Ad}(X_j)(-\epsilon_j))X_jX_j^{-1}X_i\exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\
&= X_i^{-1}\exp^{\wedge}_{\mathsf{G}}(\mathrm{Ad}(X_j)(-\epsilon_j))X_i\exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\
&= \exp^{\wedge}_{\mathsf{G}}(\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)(-\epsilon_j))X_i^{-1}X_i\exp^{\wedge}_{\mathsf{G}}(\epsilon_i) \\
&= \exp^{\wedge}_{\mathsf{G}}(\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)(-\epsilon_j))\exp^{\wedge}_{\mathsf{G}}(\epsilon_i).
\end{aligned}$$

Using Baker–Campbell–Hausdorff formula we obtain

$$\begin{aligned}
\mathcal{H}_{n+1} &= \frac{\partial}{\partial\epsilon}[\epsilon_i + \Phi(\epsilon_i)\,\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)(-\epsilon_j) + \cdots]|_{\epsilon=0} \\
&= [\overbrace{0\cdots}^{1:j-1}\ \overbrace{-\Phi(\epsilon_i)\,\mathrm{Ad}(X_i^{-1})Ad(X_j)}^{j}\ \overbrace{\cdots 0\cdots}^{j+1:i-1} \\
&\quad \underbrace{I + \frac{\partial}{\partial\epsilon}(\Phi(\epsilon_i)\,\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)(-\epsilon_j))}_{i}\ \underbrace{\cdots 0}_{i+1:n+1}\ ]|_{\epsilon=0} \\
&= [\overbrace{0\cdots}^{1:j-1}\ \overbrace{-\Phi(\epsilon_i)\,\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)}^{j}\ \overbrace{\cdots 0\cdots}^{j+1:i-1} \\
&\quad \underbrace{I - \mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)\epsilon_j\frac{\partial}{\partial\epsilon}(\Phi(\epsilon_i))}_{i}\ \underbrace{\cdots 0}_{i+1:n+1}\ ]|_{\epsilon=0} \\
&= [\overbrace{0\cdots}^{1:j-1}\ \overbrace{-\Phi(0)\,\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)}^{j}\ \overbrace{\cdots 0\cdots}^{j+1:i-1} \\
&\quad \underbrace{I}_{i}\ \underbrace{\cdots 0}_{i+1:n+1}\ ] \\
\Phi(0) &= \sum_{n=0}^{\infty}\frac{B_n\,\mathrm{ad}(0)^n}{n!} = \frac{B_0\,\mathrm{ad}(0)^0}{0!} + 0 = 1.
\end{aligned}$$

Finally, we have

$$\mathcal{H}_{n+1} = [\overbrace{0\cdots}^{1:j-1}\ \overbrace{-\mathrm{Ad}(X_i^{-1})\,\mathrm{Ad}(X_j)}^{j}\ \underbrace{\cdots 0\cdots}_{j+1:i-1}\ \underbrace{I}_{i}\ \underbrace{\cdots 0}_{i+1:n+1}\ ]. \tag{69}$$

Acknowledgements

This work has been supported from the Unity Through Knowledge Fund (no. 24/15) under the project Cooperative Cloud based Simultaneous Localization and Mapping in Dynamic Environments (cloudSLAM) and has been carried out within the activities of the Centre of Research Excellence for Data Science and Cooperative Systems supported by the Ministry of Science and Education of the Republic of Croatia under Grant 533-19-15-0007.

Notes

1. http://vision.in.tum.de/data/datasets/rgbd-dataset/tools#evaluation
2. http://cvlibs.net/datasets/kitti/eval_odometry.php
3. The Euclidean space can be formed only by employing direct product, while other ways to concatenate Lie groups also exist, i.e., semi-direct product, twisted product etc.

References

Agarwal S, Mierle K and Others (2010) Ceres solver. http://ceres-solver.org.

Arrigoni F, Rossi B and Fusiello A (2016) Spectral synchronization of multiple views in se(3). *SIAM Journal on Imaging Sciences* 9(4): 1963–1990.

Aulinas J, Petillot Y, Salvi J and Lladó X (2008) The SLAM Problem: A Survey. In: *Conference on Artificial Intelligence Research and Development*. IOS Press, pp. 363–371.

Bailey T and Durrant-Whyte H (2006) Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics & Automation Magazine* 13(3): 108–117.

Barfoot TD and Furgale PT (2014) Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics* 30(3): 679–693.

Barrau A and Bonnabel S (2015) Intrinsic filtering on Lie groups with applications to attitude estimation. *IEEE Transactions on Automatic Control* 60(2): 436–449.

Bell BM (1994) The iterated kalman smoother as a gauss–newton method. *SIAM Journal on Optimization* 4(3): 626–636.

Bell BM and Cathey FW (1993) The Iterated Kalman Filter Update as a Gauss-Newton Method. *IEEE Transactions on Automatic Control* 38(2): 294–297. DOI:10.1109/9.250476.

Bourmaud G and Mégret R (2015) Robust large scale monocular visual SLAM. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 1638–1647.

Bourmaud G, Mégret R, Arnaudon M and Giremus A (2015) Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions. *Journal of Mathematical Imaging and Vision* 51(1): 209–228. DOI: 10.1007/s10851-014-0517-0.

Bourmaud G, Mégret R, Giremus A and Berthoumieu Y (2013) Discrete Extended Kalman Filter on Lie groups. In: *European Signal Processing Conference (EUSIPCO)*. pp. 1–5.

Bourmaud G, Megret R, Giremus A and Berthoumieu Y (2016) From Intrinsic Optimization to Iterated Extended Kalman Filtering on Lie Groups. *Journal of Mathematical Imaging and Vision* 55(3): 284–303.

Briales J and Gonzalez-Jimenez J (2016) Fast global optimality verification in 3d slam. In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, pp. 4630–4636.

Briales J and Gonzalez-Jimenez J (2017) Cartan-sync: Fast and global se(d)-synchronization. *IEEE Robotics and Automation Letters* 2(4): 2127–2134.

Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, Achtelik MW and Siegwart R (2016) The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research* 35(10): 1157–1163.

Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I and Leonard JJ (2016) Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics* 32(6): 1309–1332.

Carlevaris-Bianco N, Kaess M and Eustice RM (2014) Generic Node Removal for Factor-Graph SLAM. *IEEE Transactions on Robotics* 30(6): 1371–1385.

Carlone L (2013) A convergence analysis for pose graph optimization via gauss-newton methods. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 965–972.

Carlone L, Tron R, Daniilidis K and Dellaert F (2015) Initialization techniques for 3d slam: A survey on rotation estimation and its use in pose graph optimization. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 4597–4604.

Ćesić J, Marković I, Bukal M and Petrović I (2017) Extended information filter on matrix Lie groups. *Automatica* 82: 226–234.

Chirikjian GS (2012) *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer.

Chow CK and Liu CN (1968) Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory* 14(3): 462–467.

Civera J, Grasa OG, Davison AJ and Montiel JMM (2010) 1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry. *Journal of Field Robotics* 27(5): 609–631.

Cvišić I and Petrović I (2015) Stereo odometry based on careful feature selection and tracking. In: *European Conference on Mobile Robots (ECMR)*. pp. 0–5.

Davis T (2006) *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics.

Davison AJ, Reid ID, Molton ND and Stasse O (2007) MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(6): 1052–1067.

Dellaert F (2012) Factor Graphs and GTSAM: A Hands-on Introduction. Technical report, GT RIM.

Dellaert F and Kaess M (2006) Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *The International Journal of Robotics Research* 25(12): 1181–1203.

Dissanayake G, Huang S, Wang Z and Ranasinghe R (2011) A review of recent developments in Simultaneous Localization and Mapping. In: *Industrial and Information Systems*. pp. 477–482.

Dissanayake MWMG, Newman P, Clark S, Durrant-Whyte HF and Csorba M (2001) A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation* 17(3): 229–241.

Durrant-Whyte H and Bailey T (2006) Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine* 13(2): 99–110.

Eade E (2008) *Monocular Simultaneous Localisation and Mapping*. PhD Thesis, University of Cambridge.

Eade E, Fong P, Munich ME and Robotics E (2010) Monocular Graph SLAM with Complexity Reduction. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 3017–3024.

Engel J, Schops T and Cremers D (2014) *LSD-SLAM: Large-Scale Direct Monocular SLAM*. Springer International Publishing.

Engel J, Stueckler J and Cremers D (2015) Large-Scale Direct SLAM with Stereo Cameras. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1935–1942.

Eustice RM, Singh H and Leonard JJ (2006) Exactly Sparse Delayed-State Filters for View-Based SLAM. *IEEE Transactions on Robotics* 22(6): 1100–1114.

Geiger A, Lenz P and Urtasun R (2012) Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 3354–3361.

Gilitschenski I, Kurz G, Julier SJ and Hanebeck UD (2015) Unscented orientation estimation based on the Bingham distribution. *IEEE Transactions on Automatic Control* in press: 11.

Glover J and Kaelbling LP (2013) Tracking 3-D rotations with the quaternion Bingham filter. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT).

Grisetti G, Kummerle R, Stachniss C and Burgard W (2010) A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine* 2(4): 31–43.

Guennebaud G, Jacob B et al. (2010) Eigen v3. http://eigen.tuxfamily.org.

Guivant J and Nebot E (2001) Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementation. *IEEE Transactions on Robotics and Automation* 17: 242–257.

Guivant J and Nebot E (2002) Improving computational and memory requirements of simultaneous localization and map building algorithms. *IEEE International Conference on Robotics and Automation (ICRA)* 3: 2731–2736.

Hertzberg C, Wagner R, Frese U and Schröder L (2013) Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion* 14(1): 57–77.

Julier S and Uhlmann J (2004) Unscented Filtering and Non Linear Estimation. *Proceedings of the IEEE* 92(3): 401–422.

Kaess M, Johannsson H, Roberts R, Ila V, Leonard JJ and Dellaert F (2012) iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research* 31(2): 216–235.

Kaess M, Ranganathan A and Dellaert F (2008) iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics* 24(6): 1365–1378.

Kohlhepp P, Pozzo P, Walther M and Dillmann R (2004) Sequential 3D-SLAM for mobile action planning. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1. pp. 722–729.

Konolige K (2010) Sparse Sparse Bundle Adjustment. In: *British Machine Vision Conference*. BMVA Press, pp. 102.1–102.11.

Konolige K, Grisetti G, Kümmerle R, Burgard W, Limketkai B and Vincent R (2010) Efficient Sparse Pose Adjustment for 2D mapping. In: *EEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 22–29.

Kretzschmar H and Stachniss C (2012) Information-theoretic compression of pose graphs for laser-based SLAM. *The International Journal of Robotics Research* 31(11): 1219–1230.

Kuemmerle R, Grisetti G, Strasdat H, Konolige K and Burgard W (2011) g2o: A General Framework for Graph Optimization. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, pp. 3607–3613.

Lenac K, Ćesić J, Marković I, Cvišić I and Petrović I (2017) Revival of filtering based SLAM? Exactly sparse delayed state filter on Lie groups. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 2012–1018.

Montemerlo M, Thrun S, Koller D and Wegbreit B (2002) FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: *National Conference on Artificial Intelligence (AAAI)*. AAAI, pp. 593–598.

Montemerlo M, Thrun S, Roller D and Wegbreit B (2003) FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 1151–1156.

Mourikis AI and Roumeliotis SI (2007) A multi-state constraint Kalman filter for vision-aided inertial navigation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 3565–3572.

Mur-Artal R, Montiel JMM and Tardós JD (2015) ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31(5): 1147–1163.

Mur-Artal R and Tardós JD (2017) ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics* PP(99): 1–8.

Nikolic J, Rehder J, Burri M, Gohl P, Leutenegger S, Furgale PT and Siegwart R (2014) A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 431–437.

Park W, Yunfeng Wang and Chirikjian GS (2010) The Path-of-probability Algorithm for Steering and Feedback Control of Flexible Needles. *The International Journal of Robotics Research* 29(7): 813–830.

Pire T, Fischer T, Civera J, de Cristóforis P and Jacobo-Berlles J (2015) Stereo parallel tracking and mapping for robot localization. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1373–1378.

Polok L, Ila V, Solony M, Smrz P and Zemcik P (2013) Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics. In: *Intelligent Autonomous Vehicles Symposium (IFAC)*. Berlin, Germany.

Selig JM (2005) Lie Groups and Lie Algebras in Robotics. In: *Computational Noncommutative Algebra and Applications*. Springer Science and Business Media, pp. 101–125.

Stachniss C, Hahnel D and Burgard W (2004) Exploration with active loop-closing for FastSLAM. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2. pp. 1505–1510.

Stoyanov T, Magnusson M, Andreasson H and Lilienthal AJ (2012) Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The International Journal of Robotics Research* 31(12): 1377–1393.

Tardós JD, Neira J, Newman PM and Leonard JJ (2002) Robust Mapping and Localization in Indoor Environments Using Sonar Data. *The International Journal of Robotics Research* 21(4): 311–330.

Thrun S, Liu Y, Koller D, Ng AY, Ghahramani Z and Durrant-Whyte H (2004) Simultaneous Localization and Mapping With Sparse Extended Information Filters. *The International Journal of Robotics Research* 23(7-8): 693–716.

Walter M, Eustice R and Leonard J (2007) *A Provably Consistent Method for Imposing Sparsity in Feature-Based SLAM Information Filters*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 214–234.

Wang Y and Chirikjian GS (2008) Nonparametric Second-Order Theory of Error Propagation on Motion Groups. *The International Journal of Robotics Research* 27(11): 1258–1273.

Wang Z, Huang S and Dissanayake G (2006) *Implementation Issues and Experimental Evaluation of D-SLAM*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 155–166. DOI: 10.1007/978-3-540-33453-8_14.

Weingarten J and Siegwart R (2006) 3D SLAM using planar segments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 3062–3067.