Revival of filtering based SLAM? Exactly sparse delayed state filter on Lie groups

Kruno Lenac, Josip Ćesić, Ivan Marković, Igor Cvišić, Ivan Petrović*

***Abstract*— Simultaneous Localization And Mapping (SLAM) is a core element of every modern mobile autonomous robot. The underlying engine of a SLAM system is its back-end, which aims at optimally estimating trajectory and the map of the robot's environment based on sensor data abstractions. Over the past 10 years filtering based SLAM solutions gave in to graph optimization approaches, since the latter dominated in performance over a wider range of applications. In this paper we propose a novel SLAM back-end based on the exactly sparse delayed state filter (ESDSF) and the extended Kalman filter on Lie groups (LG-EKF). Using LG-EKF directly would yield to same limitations as the early EKF-based SLAM approaches; therefore, we derive the ESDSF on Lie groups, which we dub LG-ESDSF. The proposed filter retains all the good characteristics of the classic ESDSF, but also respects the state space geometry by employing filtering equations directly on Lie groups. We have compared our SLAM system with two current state-of-the-art SLAM solutions, namely ORB-SLAM and LSD-SLAM, on the KITTI vision benchmark suite. Results showed that the proposed SLAM based on the LG-ESDSF back-end can match and even outperform methods based on graph optimization techniques.**

I. Introduction

Simultaneous Localization And Mapping (SLAM) has become a core element of every modern autonomous mobile robot. It gives a mobile robot the ability to map a new unknown environments and localize itself within at the same time. The SLAM system can usually be divided in two functional components: (i) SLAM *front-end*, which takes care of low-level sensor data processing, i.e., sensor data abstracting, and (ii) SLAM *back-end*, which based on the front-end data abstractions takes care of pose estimation based on adequate handling of pose constraints. In the past 20 years numerous solutions for both SLAM back-end and SLAM front-end have been introduced [1]. SLAM back-end algorithms, which is the focus of the present paper, can be divided in two main groups depending on the view to state optimization: (i) filtering approaches, and (ii) graph optimization approaches.

First solutions to the SLAM problem were based on filtering and employed the Extended Kalman Filter (EKF) [2] or the Particle filter [3] to predict and optimize the robot's location and the map. While revolutionary at the time, methods suffered from slow execution time and were not suitable for large-scale real-time operations. With time, faster and more accurate methods were introduced. In [4] authors proposed a novel EKF system architecture for 3D position tracking which enabled pose tracking with a 3D range image understanding system and uncertainty propagation. Work in [5] presented 3D SLAM based on EKF that uses planar segments extracted from laser range thus dramatically reducing number of features needed for accurate localization. In [6] SLAM system based on a mono camera was presented. Although it could not work in a large scale environment it used a sparse map of landmarks that enabled it to work over long periods with a high frequency (30 Hz). The SLAM solution presented in [7] used a combination of EKF and Random Sample Consensus (RANSAC) that relied on prior probabilistic information from the EKF in the RANSAC hypothesis generation. This significantly increased processing speed without the loss of discriminative power.

In the last 10 years the focus has shifted from filtering SLAM approaches to graph optimization approaches. One of the first successful alternatives to EKF SLAM, dubbed $\sqrt{\text{SLAM}}$, was presented in [8]. $\sqrt{\text{SLAM}}$ used a smoothing approach to solve the SLAM problem and achieved better performance in both computation time and accuracy than contemporary existing EKF SLAM solutions. In [9] authors presented a method for optimizing large pose graphs called the sparse pose adjustment (SPA) which is similar to $\sqrt{\text{SLAM}}$ with the main difference in efficient construction of the linear subproblem, by employing ordered data structures, and in using the Levenberg–Marquardt (LM) algorithm instead of the standard nonlinear least-square method. Graph optimization SLAM solution presented in [10] used 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. Graph based SLAM solution is also presented in [11]. In order to speed up computation and allow execution in large scale environments, authors divided global graph into subgraphs which are optimized independently using LM algorithm. Subgraphs are then matched and combined into global graph using loopy belief propagation algorithm called large scale relative similarity averaging.

Current state-of-the-art method for solving SLAM problem by graph optimization, called g^{2}o, was presented in [12]. It represents a general framework for performing optimization of nonlinear least squares problems that can be represented as a graph. By utilizing sparsity in the graph together with the advanced methods to solve sparse systems and special structures that often occur in the SLAM

This work has been supported from the Unity Through Knowledge Fund under the project *Cooperative Cloud based Simultaneous Localization and Mapping in Dynamic Environments* (cloudSLAM) and by the Ministry of Science, Education and Sports of the Republic of Croatia under the grant *Center of Research Excellence for Data Science and Cooperative Systems* (CoE ACROSS).

*Authors are with the University of Zagreb Faculty of Electrical Engineering and Computing, Croatia. {kruno.lenac, josip.cesic, ivan.markovic, igor.cvisic, ivan.petrovic}@fer.hr

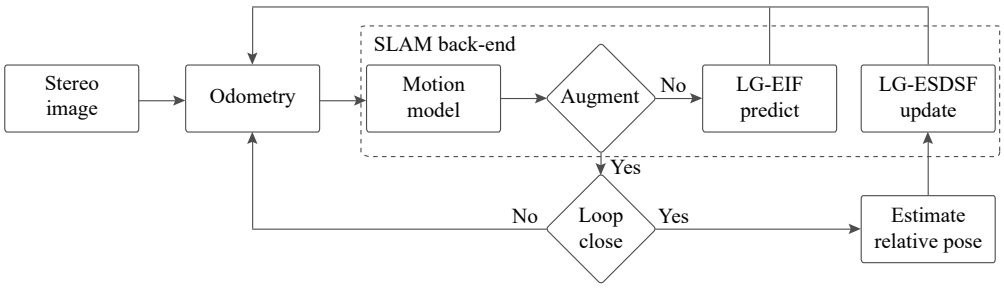

Fig. 1: Layout of an ESDSF SLAM system

graphs, g^{2}o SLAM back-end outperformed all other existing solutions. Two current state-of-the-art SLAM solutions, namely ORB SLAM and LSD-SLAM, both use g^{2}o for SLAM back-end. LSD-SLAM was first presented for mono-cameras in [13] and afterwards a stereo-solution [14] was introduced. It employs a direct and featureless method which minimizes photometric errors between images in order to estimate robot location. 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 [15] and later for stereo and RGB-D cameras [16]. It uses ORB features for mapping, loop closing and tracking and employs covisibility graph and survival of the fittest strategy to allow real-time execution over long periods in large-scale environments.

Although it would seem that the SLAM back-end optimization problem is solved and that filtering approaches in general offer worse performance than the graph optimization, in this paper we assert that this may not be the case. One of the main advantages of the graph optimization methods over filtering approaches is in their ability to respect geometry of the state space of both the trajectory and landmark poses. For example, g^{2}o respects the geometry of the state space by representing the states on Lie groups and performs optimization in the pertaining Lie algebra. On the other hand, filtering approaches in the past used suboptimal representations of the state space geometry which is one of the main reason they were unable to compete with the newly developed graph-based optimization SLAM solutions. Only recently, the first complete solutions to the EKF [17], [18] and unscented Kalman filter [19] on Lie groups have been introduced, which we designate as LG-EKF and LG-UKF, respectively. In this paper we show that by implementing the SLAM back-end based on a filtering approach using Lie groups one can attain state-of-the-art performance comparable to graph optimization approaches.

We propose a novel filtering SLAM back-end solution that respects the state space geometry by representing it as a Lie group. The proposed approach is based on the framework of the exactly sparse delayed state filter (ESDSF) [20] and LG-EKF [18]. ESDSF is a special form of the Extended Information Filter (EIF) whose main advantage is sparsity of the information matrix. LG-EKF is an extension of the EKF where the state is defined as a random variable residing on a Lie group. By using the LG-EKF one ensures that the filtering equations intrinsically take the nonlinear geometry of the state space into account and that pertaining uncertainties are handled correctly. However, LG-EKF suffers from the same limitations as the standard Euclidean EKF when it comes to its application to SLAM problems. The main contribution of this paper is the introduction of a novel ESDSF on Lie groups (LG-ESDSF) which retains all the good characteristics of the classic ESDSF implementation, but also respects the state space geometry by employing filtering equations respecting Lie groups. In the end, we show on a popular public dataset that the proposed LG-ESDSF SLAM back-end can match and even outperform state-of-the-art SLAM solutions based on graph optimization techniques.

II. Euclidean ESDSF SLAM

In this section we briefly present the fundamentals of the Euclidean ESDSF as a SLAM back-end. The SLAM back-end based on the ESDSF is a so-called pose graph SLAM which means that states of the filter are represented by a discrete robot trajectory. Map consists of measurements assigned to each of the discrete trajectory states which is why map landmarks are dependent only on the pertaining state and are not correlated to each other. This allows independent estimation of the trajectory and environment map. Since map estimation is not directly dependent on the SLAM back-end, in the present paper we do not discuss the mapping components.

The key components of an ESDSF SLAM system based on, e.g., stereo vision, are displayed in Fig. 1. There we can see that all blocks except the SLAM back-end block belong to the SLAM front-end. All modules within the SLAM back-end are responsible for robot localization and trajectory building. Trajectory consists of discrete states X_i, $i = 1, \ldots, n$, where each state is represented by a robot's pose (3D position and orientation) expressed in the coordinate frame assigned to the initial state. Whenever new odometry data becomes available, motion model is used to predict the latest stored pose X_n to $\bar{X}_n$. After that, the trajectory is augmented with $\bar{X}_n$ and marginalized by X_n. If the pose difference between X_{n-1} and $\bar{X}_n$ exceeds a predefined threshold, the latest predicted pose $\bar{X}_n$ is added to the trajectory and designated as X_{n+1}.

Sensor measurements which were taken at the moment when X_{n+1} was added are stored and *loop detection* algorithm begins to search for possible loop closings between the newly added state X_{n+1} and other states which were already stored in the trajectory. If loop closing is detected between states X_{n+1} and X_j, a *relative pose estimation* (RPE) algorithm estimates the relative transform between states X_{n+1} and X_j. This result is then reported back to the SLAM back-end, which incorporates it as a pose constraint into the pose graph and performs trajectory update. Regardless of the augmentation or loop detection, the back-end continuously predicts the robot's pose in the background using odometry information. Given this overview of the pose graph construction, in the sequel we focus on the SLAM back-end based on ESDSF.

In the case of ESDSF, robot trajectory T_n, consisting of n pose samples or states, is a Gaussian random variable

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

where μ_{X_i} and Σ_{X_i} represent mean and covariance of a pose sample 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 $\eta = \Lambda\mu$ and $\Lambda = \Sigma^{-1}$. Each state X_i consists of a robot position and orientation at the time of augmentation. Depending on the chosen orientation representation, in the case of the Euclidean ESDSF it can hold 6 elements in the case of Euler angles or 7 elements in the case of quaternions.

Motion based model is described as a nonlinear first order Markov process

$$X_{n+1} = f(X_n, \Omega_n, w_n), \tag{2}$$

where the control signals Ω_n consist of changes in the robot pose at time step n as obtained from odometry, while w_n represents zero-mean white Gaussian noise with covariance Q_n. As shown in [20], when using a first order Markov process motion model (2), the trajectory information matrix has a sparse tridiagonal structure

$$\Lambda_n = \begin{bmatrix} \Lambda_{X_{n,n}} & \Lambda_{X_{n,n-1}} & \\ \Lambda_{X_{n-1,n}} & \Lambda_{X_{n-1,n-1}} & \Lambda_{X_{n-1,n-2}} \\ & \Lambda_{X_{n-2,n-1}} & \Lambda_{X_{n-2,n-2}} \\ & \vdots & \vdots \end{bmatrix}. \tag{3}$$

This is the key advantage of the ESDSF which enables fast computation of the matrix inverse using specially designed sparse-matrix solvers.

In ESDSF, the prediction step is defined as augmentation of T_n with $\bar{X}_n$, followed by marginalization of X_n

$$\begin{aligned} p(\bar{X}_n, X_{1:n-1} | y_{1:m}, \Omega_{1:n}) &= \mathcal{N}(\bar{\mu}_n, \bar{\Sigma}_n) = \mathcal{N}(\bar{\eta}_n, \bar{\Lambda}_n) \\ &= \int p(\bar{X}_n, X_n, X_{1:n-1} | y_{1:m}, \Omega_{1:n}) \mathrm{d}X_n \,, \end{aligned} \tag{4}$$

where $y_{1:m}$ and $\Omega_{1:n}$ represent history of all measurements and odometry data up to and including time step n. Once conditions for finally adding the state to the trajectory are fulfilled, $\bar{X}_n$ becomes X_{n+1} and the process continues. Augmenting the trajectory T_n with a new state X_{n+1}, i.e., $T_{n+1} = [X_{n+1} \;\; T_n]^{\mathrm{T}}$ requires only the addition of three new blocks to the information matrix: $\Lambda_{X_{n+1,n+1}}$, $\Lambda_{X_{n+1,n}}$ and $\Lambda_{X_{n,n+1}}$. All the other elements remain unchanged and sparsity is preserved. Similarly for marginalization, since X_n is connected to states X_{n+1} and X_{n-1}, only four blocks of the information matrix Λ_n need to be changed: $\Lambda_{X_{n,n}}, \Lambda_{X_{n,n-1}}, \Lambda_{X_{n-1,n}},$ and $\Lambda_{X_{n-1,n-1}}$.

Measurement model in the ESDSF SLAM system is given in the form of a relative pose between states X_i and X_j. The update step in ESDSF is constant time as it affects only blocks sharing information associated to X_i and X_j. This is because Jacobian H_{n+1} of the measurement function is given as

$$\begin{aligned} y_{n+1} &= h(X_i, X_j) + v_{n+1},\; v_{n+1} \sim \mathcal{N}(0, P_{ij}) \\ H_{n+1} &= \begin{bmatrix} \cdots & 0 & \cdots & \frac{\partial h}{\partial X_i} & \cdots & \frac{\partial h}{\partial X_j} & \cdots \end{bmatrix}. \end{aligned} \tag{5}$$

The matrix H_{n+1} is always sparse, since y_{n+1} depends only on X_i and X_j. When the measurement arrives at the time stamp $n+1$, relative pose error v_n is determined from the RPE module. The error v_n than acts as measurement noise and is assumed to be a white zero-mean Gaussian with covariance matrix P_{ij}.

III. Lie group and algebra preliminaries

We now introduce the necessary prerequisites for derivation of the ESDSF on Lie groups. Group operators, composition and inversion, are smooth operations, given simply as matrix multiplication and inversion. Lie algebra $\mathfrak{g}$ elements represent a tangent space of a group G at the identity element [21]. In particular, a Lie algebra is an open neighbourhood around $\mathbf{0}^p$ in the tangent space of G at the identity I. The matrix exponential and logarithm establish a local diffeomorphism given as

$$\exp_{\mathsf{G}} : \mathfrak{g} \to \mathsf{G} \quad \text{and} \quad \log_{\mathsf{G}} : \mathsf{G} \to \mathfrak{g}\,. \tag{6}$$

The Lie algebra $\mathfrak{g}$ associated to a p-dimensional matrix Lie group $\mathsf{G} \subset \mathbb{R}^{n \times n}$ is a p-dimensional vector space defined by a basis consisting of p real matrices E_r, $r = 1, .., p$, often referred to as generators [22]. A linear isomorphism between $\mathfrak{g}$ and $\mathbb{R}^p$ is given by

$$[\cdot]^{\vee}_{\mathsf{G}} : \mathfrak{g} \to \mathbb{R}^p \quad \text{and} \quad [\cdot]^{\wedge}_{\mathsf{G}} : \mathbb{R}^p \to \mathfrak{g}\,. \tag{7}$$

For brevity, we will use the following notation [23]

$$\exp^{\wedge}_{\mathsf{G}}(x) = \exp_{\mathsf{G}}([x]^{\wedge}_{\mathsf{G}}) \quad \text{and} \quad \log^{\vee}_{\mathsf{G}}(X) = [\log_{\mathsf{G}}(X)]^{\vee}_{\mathsf{G}}\,, \tag{8}$$

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, denoted as Ad_{G} and ad_{G}. A More detailed discussion on these concepts and the used notation can be found in [24].

To make use of ESDSF on Lie groups we need to first establish an error distribution. If $\epsilon \triangleq \log_{\mathsf{G}}^{\vee}(X^I)$ is tightly focused around the identity element X^I, it can be described with a Euclidean Gaussian $\epsilon \sim \mathcal{N}_{\mathbb{R}^p}(\mathbf{0}^{p\times 1}, P)$ [25], and we say X^I follows a concentrated Gaussian distribution (CGD) on G around the identity [18]. This distribution can then be translated over G by using the left action of the Lie group, and finally the random variable X is as follows

$$X = \mu \exp_{\mathsf{G}}^{\wedge}(\epsilon) \ , \text{with} \ \ X \sim \mathcal{G}(\mu, P)\,, \tag{9}$$

where $\mathcal{G}$ denotes the CGD [25]. For the present SLAM, we particularly employ the special Euclidean group $\mathsf{SE}(3)$ and the aforementioned operators can be found in [26].

IV. Lie group ESDSF SLAM

The proposed ESDSF filter on Lie groups is based on the LG-EKF framework presented in [18]. However, in order to derive the ESDSF on Lie groups, there are several ingredients that need to be solved. First we need to have an information form of the filter on Lie groups, and second, we need to be able to compute the augmentation and marginalization of a CGD. The information form was presented in our previous work [26] where we proposed the extended information filter on Lie groups (LG-EIF). The augmentation and marginalization of a CGD was presented in [23] in the context of computing the prediction step of the iterated extended Kalman filter on Lie groups. The idea therein was to approximate a CGD product with a joint CGD, after which one of the variables is marginalized out. The procedure needed to solve this problem yielded results that can be used in the prediction step of our LG-ESDSF. In the sequel, we present modifications of each of the ESDSF steps that are required for derivation of the LG-ESDSF.

A. Motion model

In LG-ESDSF each discrete pose X_i of the robot's trajectory is represented by an $\mathsf{SE}(3)$ group element

$$X_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}, X_i \sim \ \mathcal{N}(\mu_{X_i}, \Sigma_{X_{i,i}})\,, \tag{10}$$

where R_i represents a 3×3 rotation matrix defining the robot's orientation in the global frame and $t_i = [x_i, y_i, z_i]$ represents robot's position in the global frame. Trajectory T_n is no longer a vector but a block diagonal $n \times n$ matrix

$$T_n = \begin{bmatrix} X_1 & 0 & 0 & 0 \\ 0 & X_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & X_n \end{bmatrix}, T_n \sim \ \mathcal{N}(\eta_n, \Lambda_n)\,. \tag{11}$$

For LG-ESDSF the trajectory is required to be on the Lie algebra $\mathfrak{se}(3)$ instead of the Lie group. For this reason we define the trajectory $\tau_n = \log_{\mathsf{G}}^{\vee}(T_n)$ as follows

$$\tau_n = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \log_{\mathsf{G}}^{\vee}(X_1) \\ \log_{\mathsf{G}}^{\vee}(X_2) \\ \vdots \\ \log_{\mathsf{G}}^{\vee}(X_n) \end{bmatrix}, \tag{12}$$

where x_i represents state X_i on the Lie algebra $\mathfrak{se}(3)$ and in this case $\mathsf{G} = \mathsf{SE}(3)$. Relation between Λ_n and τ_n is the same as in the standard ESDSF $\tau_n = \Lambda_n^{-1}\eta_n$.

In LG-ESDSF we also assume a non-linear first order Markov process, except that the motion model (2) is now defined on G

$$X_{n+1} = f(X_n, \Omega_n, w_n) = X_n \ \exp_{\mathsf{G}}^{\wedge}(\Omega_n + w_n)\ , \tag{13}$$

where in the present paper $\Omega_n = [\Delta t, \Delta r]$ represents robot's pose change between X_{n+1} and X_n. Position change is represented by $\Delta t = [\Delta x, \Delta y\, \Delta z]$, while the change in rotation Δr is represented using euler-axis convention. The prediction covariance matrix is computed as

$$\Sigma_{n+1} = \mathcal{F}_n \Sigma_n \mathcal{F}_n^{\mathrm{T}} + \Psi(\Omega_n) Q_n \Psi(\Omega_n)^{\mathrm{T}} \tag{14}$$

$$\mathcal{F}_n = \mathrm{Ad}\left(\exp_{\mathsf{G}}^{\wedge}(-\Omega_n)\right) + \Psi(\Omega_n), \tag{15}$$

where Ψ is the right Jacobian of G [27], while $\mathscr{C}_k$ denotes the linearization of the motion model (13) at X_n [18]

$$\Psi(v) = \sum_{m=0}^{\infty} \frac{(-1)^m}{(m+1)!} \mathrm{ad}(v)^m\ ,\ v \in \mathbb{R}^p\,, \tag{16}$$

$$\mathscr{C}_n = \frac{\partial}{\partial \epsilon} \Omega\left(X_n \exp_{\mathsf{G}}^{\wedge}(\epsilon)\right)|_{\epsilon=0}\,. \tag{17}$$

In the case of LG-ESDSF, $\Omega(\cdot)$ is not a function of X_n which means that (17) evaluates to zero, i.e., $\mathscr{C}_n = 0$, hence (14) evaluates to $\mathcal{F}_n = \mathrm{Ad}\left(\exp_{\mathsf{G}}^{\wedge}(-\Omega_n)\right)$. For brevity, we also introduce the following notation

$$\mathcal{Q}_n = \Psi_n Q_n \Psi_n^{\mathrm{T}}, \qquad \Psi_n = \Psi(\Omega_n)\,. \tag{18}$$

B. LG-ESDSF prediction

During the prediction step trajectory is augmented with $\bar{X}_n$ and then marginalized by X_n. Based on augmentation and marginalization results for CGDs derived in [23], [28], it can be shown that for the information form the resulting expressions for $\bar{\eta}_n$ and $\bar{\Lambda}_n$ are:

$$\bar{\eta}_n = \begin{bmatrix} \mathcal{Q}_n^{-1}\mathcal{F}_n\beta_n^{-1}\eta_{X_n} + \alpha_n(\mu_{\bar{X}_n} - \mathcal{F}_n\mu_{X_n}) \\ \eta_{X_{n-1}} - \Lambda_{mX_n}(\eta_{X_n} - \mathcal{F}_n^T\mathcal{Q}_n^{-1}(\mu_{\bar{X}_n} - \mathcal{F}_n\mu_{X_n})) \\ \eta_{X_{n-2}} \\ \vdots \end{bmatrix}$$

$$\bar{\Lambda}_n = \begin{bmatrix} \alpha_n & \mathcal{Q}_n^{-1}\mathcal{F}_n\beta_n^{-1}\Lambda_{X_{n,n-1}} & 0 \\ \Lambda_{X_{n-1,n}}\beta_n^{-1}\mathcal{F}_n^T\mathcal{Q}_n^{-1} & \gamma_n & \Lambda_{X_{n,n-1}} \\ 0 & \Lambda_{X_{n,n-1}} & \Lambda_{X_{n-1,n-1}} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

where

$$\begin{aligned} \alpha_n &= (\mathcal{Q}_n + \mathcal{F}_n\Lambda_{X_{n,n}}^{-1}\mathcal{F}_n^T)^{-1} \\ \beta_n &= (\Lambda_{X_{n,n}} + \mathcal{F}_n^T\mathcal{Q}_n^{-1}\mathcal{F}_n) \\ \gamma_n &= \Lambda_{X_{n-1,n-1}} - \Lambda_{X_{n-1,n}}\beta_n^{-1}\Lambda_{X_{n,n-1}}\,. \end{aligned} \tag{19}$$

Once the threshold is exceeded we add the predicted state $\bar{X}_n$ to trajectory T_n, which now become X_{n+1} and T_{n+1},

respectively, while parameters Λ_{n+1} and η_{n+1} evaluate to

$$\eta_{n+1} = \begin{bmatrix} \mathcal{Q}_n^{-1}(\mu_{X_{n+1}} - F\mu_{X_n}) \\ \eta_{X_n} - \mathcal{F}_n^T \mathcal{Q}_n^{-1}(\mu_{X_{n+1}} - \mu_{X_n}) \\ \eta_{X_{n-1}} \\ \vdots \end{bmatrix}$$

$$\Lambda_{n+1} = \begin{bmatrix} \mathcal{Q}_n^{-1} & -Q_n^{-1}\mathcal{F}_n & 0 \\ -\mathcal{F}_n^T Q_n^{-1} & \Lambda_{X_{n,n}} + \mathcal{F}_n^T \mathcal{Q}_n^{-1}\mathcal{F}_n & \Lambda_{X_{n,n-1}} \\ 0 & \Lambda_{X_{n-1,n}} & \Lambda_{X_{n-1,n-1}} \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

C. LG-ESDSF update

Update in LG-ESDSF SLAM occurs at the same time as in ESDSF SLAM, i.e., whenever loop closing is detected between any two states X_i and X_j and RPE sends relative pose measurement. In the case of LG-ESDSF SLAM the innovation term is defined as

$$z_{n+1} = \log_{\mathsf{G}}^{\vee}\left(h\left(T_{n+1}\right)^{-1} Z_{n+1}\right), \tag{20}$$

where $h\left(T_{n+1}\right)$ represents estimate of the relative pose between states X_{n+1} and X_j acquired from the current trajectory T_{n+1} and Z_{n+1} represents RPE measurement in a form of $\mathsf{SE}(3)$ element. For calculating updated estimates of the information matrix Λ_{n+1} and the information vector η_{n+1} we can use LG-EIF update equations from [26]

$$\begin{aligned} \eta_{n+1}^{-} &= \mathcal{H}_{n+1}^{\mathrm{T}} R_{n+1}^{-1} z_{n+1} \\ \Lambda_{n+1}^{-} &= \Lambda_{n+1} + \mathcal{H}_{n+1}^{\mathrm{T}} R_{n+1}^{-1} \mathcal{H}_{n+1}\,, \end{aligned} \tag{21}$$

where R_{n+1} represents measurement uncertainty reported by RPE and matrix $\mathcal{H}_{n+1}$ is calculated as follows [18]

$$\mathcal{H}_{n+1} = \frac{\partial}{\partial \epsilon}[\log_{\mathsf{G}}^{\vee}(h(\mathrm{T}_{n+1})^{-1} h\left(\mathrm{T}_{n+1} \exp_{\mathsf{G}}^{\wedge}(\epsilon)\right)]|_{\epsilon=0}\,. \tag{22}$$

For the $\mathsf{SE}(3)$ group calculating the relative pose between states X_{n+1} and X_j reduces to simple matrix inverse and multiplication. Given that, measurement model is defined by

$$h(T_{n+1}) = X_j^{-1} X_{n+1}, \tag{23}$$

which is equal to the one used in [23] for relative pose averaging wherein (22) was shown to evaluate to

$$\mathcal{H}_{n+1} = [\cdots 0_{6\times 6} \cdots - \operatorname{Ad}(X_{n+1}^{-1}) \operatorname{Ad}(X_j) \cdots I_{6\times 6}]\,.$$

The result shows that as in the Euclidean ESDSF, $\mathcal{H}_{n+1}$ in LG-ESDSF remains a sparse matrix consisting of n 6×6 blocks of which only the blocks $(n+1)$ and j are different than $0_{6\times 6}$. However, as explained in [26] this does not complete the update step of LG-ESDSF since now the value of $\eta_{n+1}^{-} = (\Lambda_{n+1}^{-})^{-1}\eta_{n+1}^{-}$, is in general different from the zero vector, and this is in contradiction to the CGD assumption. To overcome this issue, the state reparametrization is performed [18] and the final expressions are as follows

$$\tau_{n+1}^{-} = (\Lambda_{n+1}^{-})^{-1}\eta_{n+1}^{-} \tag{24}$$

$$\Lambda_{n+1}^{+} = \Psi(\tau_{n+1}^{-})^{-T} \Lambda_{n+1}^{-} \Psi(\tau_{n+1}^{-})^{-1} \tag{25}$$

$$\eta_{n+1}^{+} = \Lambda_{n+1}^{+} \log_{\mathsf{G}}^{\vee}\left(\exp_{\mathsf{G}}^{\wedge}(\Lambda_n^{-1}\eta_n) \exp_{\mathsf{G}}^{\wedge}(\tau_n^{-})\right). \tag{26}$$

Algorithm 1 LG-ESDSF SLAM back-end pseudocode

```
Set initial values of Λ_1, X_1, η_1, x_1
loop:
Get odometry data Ω_n
Perform motion model (13) to get X̄_n
Calculate matrices F_n and Q_n from (15) and (18)
if adding new state to T_n required then
    X_{n+1} ← X̄_n
    Calculate η_{n+1} and Λ_{n+1}
    if Loop closed between X_{n+1} and X_j then
        Get relative pose measurement Z_{n+1}
        Calculate innovation z_{n+1} (20)
        Do update via (21)–(26)
else
    Calculate η̄_n and Λ̄_n
```

This concludes the update step. Pseudocode of the entire LG-ESDSF SLAM back-end is given in Algorithm 1.

D. Performance analysis

The prediction step of LG-ESDSF is constant time since it always changes the same number of blocks in the information matrix. However, it requires poses x_n and x_{n+1} represented on the Lie algebra $\mathfrak{se}(3)$. State x_{n+1} can easily be calculated from X_{n+1}, but state x_n would have to be evaluated via $x_n = \Lambda_n^{-1}\eta_n$, which should be avoided due to inversion of the information matrix. For this reason we use the vector $x_n^{n+1} = [x_{n+1}, x_n]$ which stores required state estimates. Whenever a new state is calculated, it is inserted into x_n^{n+1} and the last one is discarded, thus inversion of Λ_n is not needed during prediction. Vector x_n^{n+1} is also changed after the update step is complete by extracting the last two states x_n, x_{n+1} from the newly estimated τ_{n+1}.

By examining (24) to (26) it would appear that there is an extra inversion of Λ_{n+1} in comparison to ESDSF. In both ESDSF and LG-ESDSF it takes place before the update so as to evaluate the measurement model, and after the update in order to recover the updated trajectory. In LG-ESDSF it is also required in (24), but (26) can be written as

$$\begin{aligned} \eta_{n+1}^{+} &= \Lambda_{n+1}^{+} \log_{\mathsf{G}}^{\vee}\left(\exp_{\mathsf{G}}^{\wedge}(\Lambda_n^{-1}\eta_n) \exp_{\mathsf{G}}^{\wedge}(\tau_{n+1}^{-})\right) \\ &= \Lambda_{n+1}^{+} \log_{\mathsf{G}}^{\vee}\left(\exp_{\mathsf{G}}^{\wedge}(\tau_n) \exp_{\mathsf{G}}^{\wedge}(\tau_{n+1}^{-})\right) \\ &= \Lambda_{n+1}^{+} \tau_{n+1}^{+}\,, \end{aligned} \tag{27}$$

which means that there is no need for final computation of τ_{n+1}^{+} as it is already calculated within η_{n+1}^{+}. Therefore, there are also only two inversions of the information matrix in LG-ESDSF: first in $\tau_n = \Lambda_n^{-1}\eta_n$ and the second in (24).

The last potentially time consuming calculation is the inversion of $\Psi(\tau_{n+1}^{-})$ required in (25) after the update. Although $\Psi(\tau_{n+1}^{-})$ does have the same dimension as Λ_{n+1}^{-}, it is also a sparse matrix and always remains a strictly tridiagonal block matrix (its form is not effected by update) and its inversion reduces to n inversions of a 6×6 matrix.

TABLE I: KITTI rankings of the state-of-the-art stereo vision SLAM systems at the time of writing.

Method	Transl.	Rot. [°/m]	Sensors
LG-SLAM [this]	0.82 %	0.0020	stereo cameras
ORB-SLAM2 [16]	1.15 %	0.0027	stereo cameras
S-PTAM [32]	1.19 %	0.0025	stereo cameras
S-LSD-SLAM [14]	1.20 %	0.0033	stereo cameras

V. Experimental results

We have tested the proposed LG-ESDSF SLAM on the KITTI vision benchmark suite [29] and compared the results with state-of-the-art SLAM approaches, namely ORB-SLAM and LSD-SLAM. For the SLAM front-end we have used the SOFT stereo visual odometry [30]. Although the improved version of SOFT exists (SOFT2), we did not use this version since it is still work in progress. However, we have modified SOFT to allow loop closing detections and relative pose estimations between any two pairs of stereo images. All algorithms were implemented using C++ programming language. Testing machine was a laptop with Intel Core i7@2.6 Ghz processor and 8 GB of RAM. For solving the sparse matrix equations we used the Eigen library [31].

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 KITTI dataset provides its own metric, herein referred to as relative metric, for evaluating the odometry results. This metric does not evaluate absolute errors between the ground truth and 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. Given that, in our evaluation, we also used the automatic evaluation tool available online[1] which relies on evaluation of absolute errors.

The LG-ESDSF SLAM was evaluated online on the KITTI dataset, therein dubbed LG-SLAM. The results are given in Table I which also contains results of several other evaluated state-of-the-art SLAM systems, from which we can see that LG-SLAM achieves the smallest translational and rotational error. The results and the whole table are also available online[2], and, at the time of writing, the proposed approach ranks second among the stereo vision approaches. The results of LG-SLAM using the absolute metric are presented in Table II. Here we provide results of absolute metric for LG-SLAM together with results of SOFT visual odometry, LSD-SLAM and ORB-SLAM2 for 6 KITTI sequences that contain loop closings. The results for sequence KITTI00 is depicted in Fig. 2. Given the results in Table II, it can be noted that LG-SLAM outperformes the other two SLAM approaches on 4 out of 6 sequences. In particular, LSD-SLAM shows best performance on sequences KITTI00 and KITTI02, while

[1]http://vision.in.tum.de/data/datasets/rgbd-dataset/tools#evaluation

[2]http://cvlibs.net/datasets/kitti/eval_odometry.php

TABLE II: Results of LG-ESDSF SLAM on the KITTI dataset using the absolute metric on sequences containing loop closing.

	SOFT	LG-ESDSF	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

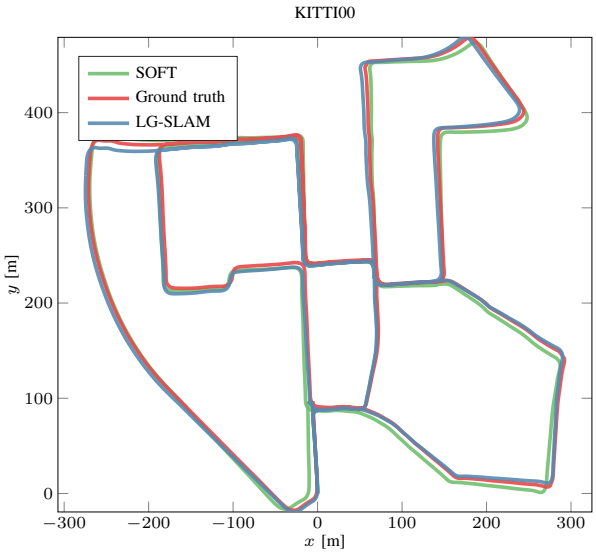

Fig. 2: LG-SLAM results on the KITTI sequence KITTI00

LG-SLAM achieves the smallest error in all the other cases. We can also see that on all of the sequences which contain loop closing, LG-SLAM significantly improves the accuracy with respect to the SOFT visual odometry.

We have also compared computation times of the proposed LG-ESDSF with the g^{2}o [12]. This was performed by simulating the LG-ESDSF on all the KITTI sequences that contain loop closings and then simulating the g^{2}o framework under the same conditions taking care that graph optimization is conducted at the same number of states and in the same order. Table III gives computation times of the LG-ESDSF update step and g^{2}o optimization. We provide maximum t_{max}, minimum t_{min} and mean t_{mean} values of all computation times together with the number of states in the trajectory at the time of the first (n_{first}) and the last (n_{last}) update/optimization. Due to the sparseness of the matrix implementation we can see that the maximum and mean computation times of the LG-ESDSF update steps are significantly lower compared to those of the g^{2}o optimization. Also, it can be noted that the minimum computation times of g^{2}o are much lower compared to its maximum computation times, which is due to fast optimization if loop closings appear frequently, whereas filter update step always takes the same time depending on the number of states n in the trajectory.

TABLE III: Minimum, maximum and mean computation times of the LG-ESDSF update step and g^2o optimization.

	t_{min} [ms]	t_{max} [ms]	t_{mean} [ms]	n_{first}	n_{last}
LG-ESDSF / g^2o					
KITTI00	11 / 22	61 / 998	35 / 349	641	1946
KITTI02	41 / 6	73 / 943	48 / 279	2040	2247
KITTI05	9 / 37	32 / 749	18 / 317	542	1094
KITTI06	9 / 23	23 / 387	13 / 221	480	625
KITTI07	7 / 179	9 / 199	7 / 194	386	391
KITTI09	18 / 432	20 / 449	18 / 441	909	913

VI. Conclusion and future work

In the last years SLAM algorithms have mostly focused on developing novel front-end methods, while the back-end has focused on graph optimization techniques and current state-of-the-art SLAM approaches are largely based on g^2o. In this paper, we have proposed a novel SLAM back-end based on the ESDSF and LG-EKF. The main contribution of the paper is derivation of the ESDSF on Lie groups. This approach enables us to respect the geometry of the state space, thus reducing localization errors in filtering methods, while keeping the prospects of the classical ESDSF. We have shown on the popular KITTI dataset that our solution can match and even outperform state-of-the-art SLAM methods in both accuracy and computation speed. As part of future work, we plan to develop reduction of the number of states by allowing marginalization of redundant states within the LG-ESDSF, thus further improving computation time.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, I. D. Reid, and J. J. Leonard, "Past , Present , and Future of Simultaneous Localization And Mapping : Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

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

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *AAAI National Conference on Artificial Intelligence*, 2002, pp. 593–598.

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

[5] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3062–3067, 2006.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[7] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.

[8] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *Int. J. Rob. Res.*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[9] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 22–29.

[10] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conference*. BMVA Press, 2010, pp. 102.1–102.11.

[11] G. Bourmaud and R. Mégret, "Robust large scale monocular visual slam," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1638–1647.

[12] R. Kümmerle, G. Grisetti, K. Rainer, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general Framework for (Hyper) Graph Optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[13] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[14] J. Engel, J. Stueckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *International Conference on Intelligent Robots and Systems (IROS)*, September 2015.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[16] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *CoRR*, vol. abs/1610.06475, 2016.

[17] G. Bourmaud, R. Mégret, A. Giremus, and Y. Berthoumieu, "Discrete extended kalman filter on lie groups," in *21st European Signal Processing Conference (EUSIPCO 2013)*, Sept 2013, pp. 1–5.

[18] G. Bourmaud, R. Mégret, M. Arnaudon, and A. Giremus, "Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 209–228, 2015.

[19] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[20] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly Sparse Extended Information Filters for Feature-based SLAM," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[21] J. M. Selig, "Lie Groups and Lie Algebras in Robotics," in *Computational Noncommutative Algebra and Applications*, 2005, pp. 101–125.

[22] W. Park, Y. Wang, and G. S. Chirikjian, "The Path-of-Probability Algorithm for Steering and Feedback Control of Flexible Needles," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 813–830, 2010.

[23] G. Bourmaud, R. Mégret, A. Giremus, and Y. Berthoumieu, "From Intrinsic Optimization to Iterated Extended Kalman Filtering on Lie Groups," *Journal of Mathematical Imaging and Vision*, vol. 55, no. 3, pp. 284–303, 2016.

[24] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer, 2012.

[25] Y. Wang and G. S. Chirikjian, "Nonparametric Second-Order Theory of Error Propagation on Motion Groups," *The International Journal of Robotics Research*, vol. 27, no. 11, pp. 1258–1273, 2008.

[26] J. Ćesić, I. Marković, M. Bukal, and I. Petrović, "Extended information filter on matrix lie groups," http://lamor.fer.hr/images/50020776/Cesic2016b.pdf, 2016 (in review).

[27] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, June 2014.

[28] G. Bourmaud, "Estimation de paramètres évoluant sur des groupes de Lie : Application à la cartographie et à la localisation d'une caméra monoculaire," Ph.D. dissertation, University of Bordeaux, 2015.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *European Conference on Mobile Robots (ECMR)*, 2015.

[31] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[32] T. Pire, T. Fischer, J. Civera, P. de Cristóforis, and J. Jacobo-Berlles, "Stereo parallel tracking and mapping for robot localization," 2015.