HAL
open science

Factor Graph-Based Smoothing Without Matrix Inversion for Highly Precise Localization

Paul Chauchat, Axel Barrau, Silvere Bonnabel

To cite this version:

Paul Chauchat, Axel Barrau, Silvere Bonnabel. Factor Graph-Based Smoothing Without Matrix Inversion for Highly Precise Localization. IEEE Transactions on Control Systems Technology, 2021, 29 (3), pp.1219 - 1232. 10.1109/TCST.2020.3001387 . hal-02943953

HAL Id: hal-02943953

https://hal.science/hal-02943953v1

Submitted on 21 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Factor graph based smoothing without matrix inversion for highly precise localization

Paul CHAUCHAT, Axel BARRAU and Silvère BONNABEL

***Abstract*—We consider the problem of localizing a manned, semi-autonomous, or autonomous vehicle in the environment using information coming from the vehicle's sensors, a problem known as navigation or simultaneous localization and mapping (SLAM) depending on the context. To infer knowledge from sensors' measurements, while drawing on a priori knowledge about the vehicle's dynamics, modern approaches solve an optimization problem to compute the most likely trajectory given all past observations, an approach known as smoothing. Improving smoothing solvers is an active field of research in the SLAM community. Most work is focused on reducing computation load by inverting the involved linear system while preserving its sparsity. The present paper raises an issue which, to the knowledge of the authors, has not been addressed yet: standard smoothing solvers require explicitly using the inverse of sensor noise covariance matrices. This means the parameters that reflect the noise magnitude must be sufficiently large for the smoother to properly function. When matrices are close to singular, which is the case when using high precision modern inertial measurement units (IMU), numerical issues necessarily arise, especially with 32-bits implementation demanded by most industrial aerospace applications. We discuss these issues and propose a solution that builds upon the Kalman filter to improve smoothing algorithms. We then leverage the results to devise a localization algorithm based on fusion of IMU and vision sensors. Successful real experiments using an actual car equipped with a tactical grade high performance IMU and a LiDAR illustrate the relevance of the approach to the field of autonomous vehicles.**

I. INTRODUCTION

State estimation is fundamental, notably for control purposes. This is particularly true for autonomous systems, such as autonomous cars. Prevailing approaches to the state estimation problem in robotics, and for inertial navigation in aerospace engineering, explicitly model sensor uncertainties, due to noise and bias, using Gaussian random variables, and then seek to compute the maximum a posteriori (MAP) state, that is the most likely state in the light of all measurements while drawing on the vehicle's dynamics. Such approaches allow combining sensor measurements optimally based on their confidence levels, as quantified by their covariance, and also allow the estimator to convey a degree of uncertainty associated to its own estimate, which may prove critical for high-level planning and low-level control of autonomous vehicles.

Paul Chauchat is with ISAE-Supaéro, University of Toulouse, 31055 Toulouse, France (e-mail: paul.chauchat@isae-supaero.fr)

Silvère Bonnabel is with MINES ParisTech, PSL Reasearch University, Centre for Robotics, 60 bd Saint-Michel, 75006 Paris, France (e-mail: silvere.bonnabel@mines-paristech.fr)

Axel Barrau is with SAFRAN TECH, Groupe Safran, Rue des Jeunes Bois - Châteaufort, 78772 Magny Les Hameaux CEDEX, France (e-mail: axel.barrau@safrangroup.com)

Simultaneous Localisation and Mapping (SLAM) has received tremendous attention in the robotics community over the past two decades. With a probabilistic approach, the corresponding mathematical problem may be formulated as a nonlinear estimation problem, that has served as a benchmark for recent developments in state estimation. The historical estimator is the extended Kalman filter, which is still state of the art of inertial navigation, but exhibits huge caveats for SLAM, because of its numerical complexity, and especially its inherent inconsistency, that prompted the use of particle filters instead [14], [21], [32], [15]. More recently, non-linear observers and filters were reintroduced for SLAM through the use of geometric frameworks, and Lie groups in particular [5], [28], [44], [43]. However, the progresses of vision sensors and vision algorithms over the past years, and their ubiquity in mobile robotics and autonomous driving, have led the community to replace particle filters with optimization based algorithms being able to return the solution to the MAP problem. Such algorithms have achieved major successes for fusion of vision and inertial measurements, see for instance, [8], [41]. They have also recently been used in GNSS applications [47], adapting them in various ways to accomodate outliers in this context [35], [2], [37]. They fall into the framework of "smoothing" where one seeks to compute the most likely entire past trajectory given sensors measurements. From a theoretical point of view, the nonlinear optimization smoothing problem has been known for decades, but solving it has long proved intractable.

Smoothing is formulated as a non-linear optimization problem which is solved through iterative methods, the most popular being the Gauss-Newton (GN), Levenberg-Marquardt, Conjugate Gradient Descent or other trust-region algorithms [11], [23], [25], [40], [24], [1] methods. Extension to non-gaussian models, also based on successive least-squares problems, were developped [39], [35], [36]. To come up with tractable algorithms, these works heavily exploit the tight link that exists between the MAP problem and factor graphs, and in particular sparse graph structures [12].

In the present paper, a shortcoming of those methods that has gone unnoticed yet to our best knowledge is raised. Indeed, all the mentioned formulations are based explicitly on the information matrix of each factor - the inverse of the covariance matrix - whose eigenvalues tend to increase as the sensors improve. But a large discrepancy between the most accurate and least accurate sensor leads to ill conditioned information matrices. This may also impact linearization errors [18], [9]. Numerical issues may thus degrade the solvers' computed solution, and in turn the state estimate accuracy and consistency.

As the cost of high performance inertial measurement units (IMU) keeps decreasing, and techological progresses are constantly made, one may anticipate that tactical grade IMUs will be used in an increasing number of autonomous systems. Their precision is so high that the resulting process noise is close to negligible as compared to other sensors. This prompts the need for solvers being robust to ill conditioning of the information matrix. Using the square-root approach of the information matrix reduces the impact of conditioning [11], [30]. Numerical issues related to it depend on the precision of the hardware used, though. Indeed, single precision implementations will much more suffer. This is noticeable as there is a need to speed up computations for real-time applications, for which single precision is generally considered [45], [10], and most industrial aerospace computers still use single precision (32 bits or less) [17]. Besides, navigation methods relying on machine learning to improve some of their bricks [7], [6], or mimicking Kalman filters [19], are on the rise, and some recent methods even rely on Gauss-Newton as one of their bricks [27]. They also could benefit from single precision algorithms to speed up their training phase.

In the present paper, we propose methods to carry out a broad class of non-linear optimization problems while avoiding issues related to ill conditioning, by deriving new exact ways of solving of the linear least-squares problems they induce. More precisely, we derive two novel linear solvers. First a first simple, robust, batch algorithm for solving least-squares based on a relevant decomposition of the problem. Then, we leverage the correspondence between Bayesian inference and least-squares in the linear case to derive a new sparsity-preserving least-squares solver. To this end, we combine Stochastic Cloning (SC) [33] and the Backward Information Forward Marginal (BIFM) [26], [42] version of the Kalman smoother. Our starting point was the finding that the Kalman filter never requires to invert process noise covariance matrices, and leads to relatively cheap computation cost, which allowed simplified implementations on hardware with precision as low as 8-bits processors [29].

These solvers are leveraged to devise a novel smoothing algorithm for localization based on IMU, vision, and additional sensors such as GNSS. The algorithm is successfully implemented on an actual car equipped with a high performance IMU and a LiDAR, and real experiments illustrate the interest of the approach in the context of highly precise motion sensors.

The paper is organized as follows. In Section II we summarize the modern factor graph based approaches to the smoothing problem of SLAM, and the issues that arise when propagation covariances approach zero. We also propose our first ("robust batch") solver. In Section III, we recall how the Kalman smoother may serve as a solver for linear least squares. However, in its standard form it does not accomodate vision measurements. To remedy this issue, we propose our second robust and sparsity-preserving solver, called "SC-BIFM", in Section IV. A simple numerical illustration of the interest of the method is presented in Section V. Finally, we demonstrate the validity of our approach in real world experiments in Section VI, in which the standard methods actually fail in single precision, while our solvers prove robust, leading to a novel accurate visual inertial localization algorithm in the presence of highly precise inertial sensors.

A. Relation to previous literature

Most of the smoothing solvers use the information form of the MAP problem, in order to take advantage of its sparsity [11], [25], [1]. It has been long recognised that using the square-root, for instance the Cholesky factorisation of the information matrix, leads to algorithms which are numerically more stable, and even enabled implementations with single precision on mobile devices [45]. This led to highly efficient incremental implementations [23], [25]. However, even square-root formulations are inherently limited by the inverses of the covariances appearing in the formulas, and there is no *transition* between zero covariances, which represent hard constraints, and numerically invertible covariances. This issue has been addressed in the signal processing community by devising new formulations of the Kalman smoother. Additionally to the well-known Modified-Bryson-Frazier Smoother [4], a new alternative has been derived, the so-called Backward Information Forward Marginals (BIFM) Smoother, which also avoids covariance inverting [26], [42]. Nevertheless, they were only expressed for acyclic graphs (i.e., without relative measurements other than the propagation, such as loop closures), which only cover a small part of the existing factor graphs. We introduce here a more general smoother, able to solve a broader class of linear least-squares, and thus applies to the non-linear optimization problems at hand in navigation, by applying the stochastic cloning method of [33] to the BIFM.

II. Factor graphs and MAP

Assume we want to track the state of a system (say, a vehicle) equipped with an IMU and/or wheel odometry, as well as a set of sensors such as cameras, LiDARs, GNSS or acoustic positioning system for UAVs. The current formulation of SLAM uses the formalism of factor graphs to cast the maximum likelihood estimation problem as a nonlinear optimization problem. Throughout the article, we will not consider landmark-based SLAM, but rather "pose-graph SLAM", where vision sensors identify loop closures and provide relative poses between key frames, triggering the back-end optimization run. Moreover, and contrary to "pure" SLAM, we allow absolute sensors such as GNSS to provide information about the vehicle, and refer to SLAM as an extended multi-sensor fusion problem, mainly for localization and navigation purposes.

A. MAP formulation of the SLAM problem

1) The state: The state is a set of all variables that characterize the vehicle and its sensors, typically its orientation, position, velocity, and all parameters of interest such as IMU biases, camera to IMU transformation parameters. The state at time k is denoted X_k. In landmark-based SLAM, the state also includes the position of the landmarks in the environment. Without loss of generality the state is assumed to evolve through a discrete-time dynamical model

$$X_{k+1} = f(X_k, u_k, w_k) \tag{1}$$

with w_k a random variable (noise) encoding model uncertainties, and u_k a measurement typically obtained by wheel speed sensors or the IMU. Generally, the initial uncertainty about the state X_0 is assumed Gaussian, that is, $X_0 \sim \mathcal{N}(\hat{X}_0, P_0)$. In the smoothing approach, one is interested in the entire trajectory, denoted

$$\chi := (X_0, X_1, \cdots, X_N). \tag{2}$$

2) The observations: We are given a set of measurements $Z_1, Z_2, \cdots, Z_K$ of the form $Z_k = h_k(X_{k_1}, \ldots, X_{k_p}, V_k)$ where each state is in $X_1, X_2, \cdots, X_k$, h_k is a known measurement function, and V_k is a random variable encoding measurements uncertainties that stem from sensors' imperfections (i.e., observations noise). The measurement function of each sensor is fixed, the subscript k indicates that different sensors may be present. In the present paper centered on inertial navigation, visual inertial odometry, and inertial pose-graph SLAM, we will consider two types of measurements:

- Observations that are made at time k and involve only the state at time k (unary observations):

$$Z_k = h_k(X_k, V_k) \tag{3}$$

- Observations at time k that involve a pair of states, that is,

$$Z_{ik} = h_k(X_i, X_k, V_k), \quad i < k \tag{4}$$

  as is the case in pose graph SLAM. These measurements may be related to the computation of relative transformations between poses at different times, typically using stereo cameras or LiDARs. They can be used as either odometry to propagate the state, or as "loop closures" [15].

For simplicity of notation we let $Z_k = h_k(\chi, V_k)$ denote both types of observations (3) and (4) at time k. This is not a problem as $X_k, X_i \in \chi$. Finally $Z = (Z_1, \cdots, Z_N)$ denotes all available measurements.

B. Maximum a posteriori (MAP) estimation

As only Z is observed, the best one can achieve is to compute the most likely state trajectory χ given all the measurements obtained, that is, maximum a posteriori (MAP) estimation. Mathematically, one seeks:

$$\chi^* = \underset{\chi}{\operatorname{argmax}}\, p(\chi \mid Z) = \underset{\chi}{\operatorname{argmax}}\, p(Z \mid \chi) p(\chi). \tag{5}$$

Assume the dynamical model (1) can be inverted with respect to the noise variable, that is, there exists a function ϕ such that

$$X_{k+1} = f(X_k, u_k, w_k) \iff w_k = \phi(X_{k+1}, X_k, u_k),$$

and similarly

$$Z_k = h_k(\chi, V_k) \iff V_k = \psi_k(\chi, Z).$$

The latter quantities define *factors* that may be viewed as constraints between variables. Indeed assuming the noises are independent variables with distribution $w_k \sim \mathcal{N}(0, Q_k)$, $V_k \sim \mathcal{N}(0, R_k)$, standard computations show the log-likelihood of the posterior writes

$$\begin{aligned} L(\chi) :&= -\log p(\chi \mid Z) = ||X_0 - \hat{X}_0||^2_{P_0} \\ &+ \sum_{k=0}^{N-1} ||\phi(X_{k+1}, X_k, u_k)||^2_{Q_k} + \sum_{k=1}^{N} ||\psi_k(\chi, Z_k)||^2_{R_k}. \end{aligned} \tag{6}$$

up to an additive constant independent from χ, where $\|e\|^2_P = e^T P^{-1} e$ is the Mahalanobis norm of the vector e given the covariance P. From (5) we see that:

$$\chi^* = (X_0^*, \cdots, X_N^*) = \underset{\chi}{\operatorname{argmin}}\, L(\chi). \tag{7}$$

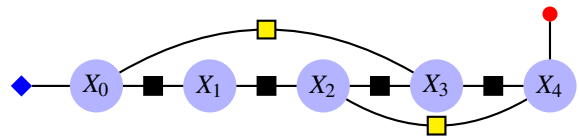


Fig. 1: MAP estimation as a factor graph: Blue circles denote the successive states of the vehicle $X_0, X_1, \cdots, X_N$ through times $0, 1 \cdots, N$. Factor nodes shown in black involve successive states and correspond to dynamical model relations (1). Factors nodes shown in orange correspond to measurements involving a pair of state variables (4). The factor nodes shown by a red circle and a blue diamond respectively represent an observation of the form (3), and a prior on the initial state.

C. Factor graph interpretation

A factor graph is a bipartite graph that contains *variable nodes* $X_j \in \chi$ and *factor nodes* η_i, each involving several variable nodes. There is an edge between a factor and each of the variables it involves. Factors may encode the model relation (1) or measurements of the form (3) or (4), and may be viewed as constraints between variables of the trajectory χ one seeks to estimate. See Figure 1 for a visualization.

D. Resolution of the nonlinear optimization problem

The minimization problem (7) is commonly tackled resorting to Gauss-Newton or Levenberg-Mardquardt methods. The idea is to start from an initial guess of the trajectory, typically obtained through dead-reckoning, and successively *linearize* the problem, and solve the obtained linear least squares, that is, a quadratic multidimensional optimization problem at each iteration. Starting from an initial guess of the trajectory $\chi^{(0)} := (X_0^{(0)}, X_1^{(0)}, \cdots, X_N^{(0)})$, the likelihood (6) may be linearized around $\chi^{(0)}$, and Gauss-Newton algorithms seek to compute

$$\delta\chi^* = \operatorname{argmin}\, \bar{L}(\delta\chi), \tag{8}$$

where $\bar{L}$ is the linearized cost. To keep the exposition simple, we assume in the sequel noises are additive, that is, (1) is of the form $X_{k+1} = f(X_k, u_k) + w_k$, and also that $Z_k = h(\chi) + V_k$.

Then, letting $\delta\chi := \chi - \chi^{(0)}$, the linearized cost writes

$$\begin{aligned}\bar{L}(\delta\chi) = &||\delta X_0 - (\hat{X}_0 - X_0^{(0)})||^2_{P_0} \\ &+ \sum_{k=0}^{N-1} ||\delta X_{k+1} - F_k \delta X_k - (f(X_k^{(0)}, u_k) - X_{k+1}^{(0)})||^2_{Q_k} \\ &+ \sum_{k=1}^{N} ||H_k \delta\chi - (Z_k - h_k(\chi^{(0)}))||^2_{R_k}\end{aligned} \tag{9}$$

where F_k denotes the Jacobian of f with respect to X_k, and H_k that of h_k at current estimate $\chi^{(0)}$. After solving (8)-(9), the linearization point is updated as $\chi^{(1)} = \chi^{(0)} + \delta\chi^*$ and serves as a new linearization point until convergence.

Remark 1: This easily generalises for smoothing on-manifold, see for instance [9], [24], [18], by simply changing the prior factor and update definitions, and adapting the jacobians accordingly. To save space, we omit it herein.

E. Resolution of the linearized optimization problem

We see at each step the algorithm is faced with the resolution of the linearized optimization problem (8)-(9). This is a standard least squares problem, and the solution comes in closed form. Note indeed that the log-likelihood $\bar{L}$ can be re-written as a single quadratic cost function:

$$\bar{L}(\delta\chi) = ||A\delta\chi - b||^2_\Sigma, \tag{10}$$

with

$$A = \begin{pmatrix} I & 0 & \dots & \dots & 0 \\ -F_1 & I & \dots & \dots & 0 \\ \dots & \dots & \dots & -F_N & I \\ & & H_1 & & \\ & & \vdots & & \\ & & H_N & & \end{pmatrix}, b = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \\ c_1 \\ \vdots \\ c_N \end{pmatrix} \tag{11}$$

where $\Sigma = \text{blkdiag}(P_0, Q, R)$, where Q and R are block diagonal matrices stacking the sequences of covariance matrices Q_n and R_k, $a_0 = \hat{X}_0 - X_0^{(p)}$, $a_i = (f(X_{i-1}^{(p)}, u_{i-1}) - X_i^{(p)})$ if $i > 0$, and $c_k = (Z_k - h_k(\chi^{(p)}))$. Computing the gradient of (10) and setting it to zero, allows one to conclude the minimizer $\delta\chi^*$ satisfies the *normal equations*:

$$(A^T\Sigma^{-1}A)\delta\chi^* = A^T\Sigma^{-1}b \tag{12}$$

Solving the normal equations (12) is the main task to be addressed when implementing factor graph optimization. Indeed, one is faced with a system of linear equations. Most popular methods rely on factorizations of the information matrix defined by $\mathscr{I} := A^T\Sigma^{-1}A$ in the form of $A^T\Sigma^{-1}A = LL^T$ with L lower triangular (Cholesky) or through a QR factorization. Matrix L is referred as the "square root" of the information matrix, and allows to solve the problem by first solving $L\eta = A^T\Sigma^{-1}b$ and then $\eta = L^T\delta\chi^*$ by back-substitution, see [11].

F. Potential issue n°1: computational complexity

A is a matrix having size $((N+1)d_x + Kd_z) \times ((N+1)d_x)$ where d_x, d_z are the dimensions of the state and observation vectors, K is the number of observations (we assumpe for simplicity all observations have identical dimension). In principle, the problem would not be tractable for localization applications. However, the sparsity of A stemming from the particularity of the problem estimation structure allows smoothing solvers to encode various ways of inverting this equation. Indeed, each factor only involves few variables (usually one or two) [16]. For instance, [23] embeds the problem in a particularly well adapted Bayes tree structure which allows for local constant-time updates at each time step by reusing previous factorizations. In a more recent work, [25] uses preconditioned conjugate gradient to invert the problem, with a high efficiency gain coming from a number of computational tricks specific to the SLAM problem. These methods usually prefer working with an equivalent formulation of (10):

$$\bar{L}(\delta X) = ||\tilde{A}\delta\chi - \tilde{b}||^2$$

with $\tilde{A} = \Sigma^{-1/2}A$ and $\tilde{b} = \Sigma^{-1/2}b$, where $\Sigma^{-1/2}$ is the square-root of Σ^{-1}. However, both formulations explicitly rely on the inverse of the covariance matrices at play.

G. Potential issue n°2: ill-conditioned normal equation matrix

The complexity issue is obvious, and much efforts in the SLAM back-end community have been devoted to it. Another potential issue that has scarcely been addressed so far, and that we bring forward in the present paper, is as follows. If the normalization matrix Σ^{-1} is ill conditioned because Σ has very small eigenvalues (or even null eigenvalues), then measurement noise is amplified and the solution may become grossly inaccurate. In the present paper, we focus on the case where dynamics (1) rely on high precision inertial sensors, which are becoming increasingly common, and lead to very small covariance matrices.

Assume the covariance matrix Q has very small eigenvalues. As $Q \to 0$, we see that $\Sigma^{-1} \to \infty$ and thus

$$\begin{aligned}(A^T\Sigma^{-1}A) &\longrightarrow \infty, \quad (A^T\Sigma^{-1}b) \longrightarrow \infty, \\ \Sigma^{-1/2}A &\longrightarrow \infty, \quad \Sigma^{-1/2}b \longrightarrow \infty.\end{aligned}$$

But this does not mean that the quantity of interest $\delta\chi^*$ degenerates in the same way, as we have an "$\frac{\infty}{\infty}$" indetermination from (12). In fact, it does converge to a finite value $\delta\chi_\infty$ which is simply the result of a lower-dimensional problem: we will prove in the next section that when $Q \to 0$ we have

$$(A^T\Sigma^{-1}A)^{-1}A^T\Sigma^{-1}b \longrightarrow \delta\chi_\infty, \tag{13}$$

Ideally, we would like the solver to be such that $\delta\chi^* \to \delta\chi_\infty$ when $Q \to 0$, and also to be able to find $\delta\chi_\infty$ even in the degenerate case where $Q = 0$. However, we cannot expect such a desirable behavior from a solver directly based on the normal equations (12). This is why we advocate in the present paper a different approach based on the Kalman smoother to solve the linearized optimization problem (8), (9).

H. A solution without matrix inversion: the robust batch solver

Assume the noise matrix of the model Q should not be inverted. Owing to the block-diagonal structure of the noise

matrix Σ, the linearized cost (10) may be split into two parts as follows:

$$\bar{L}(\delta\chi) = \|A_1\delta\chi - b_1\|^2_{\Sigma_1} + \|A_2\delta\chi - b_2\|^2_{\Sigma_2} \tag{14}$$

that is, $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$, and $\Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}$, and where Σ_1 should not be inverted, and where we require A_1 to be square and invertible. In the meantime, Σ_2 should be well-conditionned. There might be less factors with close to singular covariances than the dimension of the state, and A_1 may be completed with additional factors having well-conditioned covariances to make it square if need be. Using the associated normal equations, and the matrix inversion lemma, the minimizer of the cost (14) reads:

$$\begin{aligned} \delta\chi^* &= A_1^{-1}((I-KJ)b_1 + Kb_2), \\ K &= \Sigma_1 J^T (J\Sigma_1 J^T + \Sigma_2)^{-1}, \quad J = A_2 A_1^{-1}. \end{aligned} \tag{15}$$

The inverted term is the sum of a possibly ill-conditionned matrix with small eigenvalues, and a well-conditionned one, and should thus have no eigenvalue close to zero. Wee are now in a position to prove the result announced in equation (13).

Lemma 1: As $Q \to 0$, the solution $\delta\chi^*$ to (12) tends to a finite value.

Proof: Expanding the definitions of K and J in (15), we obtain: $\delta\chi = A_1^{-1}\left[b_1 + \Sigma_1 A_1^{-1} A_2^T (A_2 A_1^{-1}\Sigma_1 A_1^{-T} A_2^T + \Sigma_2)^{-1}(b_2 - A_2 A_1^{-1} b_1)\right]$. We see $(A_2 A_1^{-1}\Sigma_1 A_1^{-T} A_2^T + \Sigma_2) \to (A_2 A_1^{-1}\Sigma_\infty A_1^{-T} A_2^T + \Sigma_2)$ when $\Sigma_1 \to 0$, and this latter term is lower-bounded by Σ_2, so there is no indetermination in (13): δX_∞ exists. ■

This solution may become intractable as it requires the full inverse of A_1, and the size of the matrix A_1 is quadratic in the size of the trajectory. According to (11), A_1 has a lower block-triangular structure, and its inverse may be obtained analytically, but the number of linear systems to solve because of the term $J = A_2 A_1^{-1}$ still is a major caveat of this method. It is however used in Section VI as a reference for comparison purposes. In the next sections we investigate an approach based on Kalman smoothers, which reaches identical result, but with a complexity being linear in the length of the trajectory.

III. The linear Kalman smoother as a least squares solver

Consider the problem of finding the minimum of cost function $\bar{L}$ given by (9). In this section we will recall the Kalman smoother may serve as a solver to this optimization problem, albeit (for now) in the particular case of unary observations of the form (3).

A. The standard Kalman smoother

Consider the following linear system with unary observations:

$$\begin{aligned} X_{k+1} &= F_k X_k + u_k + w_k, \\ Z_k &= H_k X_k + V_k \quad \text{(unary observations)} \end{aligned} \tag{16}$$

$Z_0 = X_0 + V_0$ encodes the prior on X_0. It is easily proved that $\bar{L}$ given by (9) is related to this linear system as

$$\bar{L} = -\log p(X_0, \ldots, X_N \mid Z_1, \ldots, Z_N) + c,$$

where $c \in \mathbb{R}$ is a constant. The linear Kalman smoother computes the MAP estimate which minimizes $\bar{L}$, and thus may output the quantity of interest $\delta\chi^*$ that is required at each linearization step of the more general Gauss-Newton solver. Its implementation is generally based on a forward recursion (e.g. the Kalman filter), and a backward recursion which backpropagate the information to the past states, for instance the standard Rauch-Tung-Striebel (RTS) implementation [38].

B. The Backward Information Filter Forward Marginal (BIFM)

The RTS implementation of the Kalman Smoother avoids inverting the propagation covariance, but not the states' ones, which appear in the gain used for the backpropagation [38]. In this work, we propose to use an alternative formulation of the Kalman smoother equations, the Backward Information Filter Forward Marginal (BIFM), which specifically avoids inverting the states' covariances [26]. This approach has already been used for message passing in Gaussian factor graphs in signal processing, but only for acyclic graphs. Another version of the Kalman smoother, namely the Modified-Bryson Frasier smoother, also exists [4]. Square-root forms of both algorithms have been produced and were compared on various signal processing problems in [42]. BIFM appeared slightly superior in these cases. Moreover, it can be derived in a very "Kalman filter like" manner which does not require writing the inverse of the covariance matrices Q_k nor the inverse of the forward covariance matrix P_k defined in (17), as we are about to prove.

Consider the system (16). BIFM is based on the following approach. Suppose $P(X_k|Z_0, \cdots, Z_k) \sim \mathcal{N}(x_k^f, P_k^f)$; and likewise $P(X_k|Z_{k+1}, \cdots, Z_N) \sim \mathcal{N}(x_k^b, P_k^b)$. Here the superscripts stand for forward and backward. x_k^f is obtained by a Kalman filter. Then x_k^b may be considered as a measurement of X_k with noise covariance P_k^b, and treated as such in a Kalman update to merge it with (x_k^f, P_k^f) and obtain a final estimate $\hat{X}_k$. As there is no prior for the backward phase, that is, the prior on x_0^b is "flat", the information form of the Kalman filter must be used [30], starting with zero information prior. The detailed equations are given in the following. For the sake of readability, the forward and backward distributions are denoted by, for all k :

$$P(X_k|Z_0, \ldots, Z_k) \sim \mathcal{N}(x_k, P_k) \tag{17}$$

$$P(X_k|Z_{k+1}, \ldots, Z_N) \sim \mathcal{N}(J_k^{-1} y_k, J_k^{-1}), \tag{18}$$

where y_k and J_k are the information vector and matrix respectively.

Remark 2: Note that, here, the information matrix J_k is the inverse of a covariance, while the information matrix referred to in Section II-E, $\mathscr{I}$, is the Fisher information matrix.

The forward recursion may be performed based on the standard Kalman filtering belief. The parameters of the Gaussian are returned by the standard linear Kalman filter by alternating between the propagation step:

$$\begin{aligned} x_{k+1|k} &= F_k x_k + u_k \\ P_{k+1|k} &= F_k P_k F_k^T + Q_k \end{aligned} \tag{19}$$

and letting $K_{k+1} := P_{k+1|k}H_k^T(H_kP_{k+1|k}H_k^T + R_k)^{-1}$, the update step

$$\begin{aligned} x_{k+1} &= x_{k+1|k} + K_{k+1}(Z_k - H_k x_{k+1|k}) \\ P_{k+1} &= P_{k+1|k} - K_{k+1}H_kP_{k+1|k}. \end{aligned} \tag{20}$$

We see that even if some or even all the eigenvalues of P_0 and Q are null, the computation may be performed as long as R is non-singular. In particular, this means that the measurement noise must be sufficiently large for BIFM to be applied. As a consequence, Q can be as small as desired without leading to numerical issues. The equations for J_k and y_k in the backward recursion are obtained by setting J_N to be zero, and then considering (16) in reverse time, and combining it with (19) and (20), using the identities $x_k = J_k^{-1}y_k$, $P_k = J_k^{-1}$. The update in information form simply becomes:

$$\begin{aligned} J_{k|k} &= J_k + H_k^T R_k^{-1} H_k, \\ y_{k|k} &= y_k + H_k^T R_k^{-1} Z_k \end{aligned} \tag{21}$$

The backpropagation equations read:

$$\begin{aligned} J_k &= F_k^T(I + J_{k+1|k+1}Q_k)^{-1}J_{k+1|k+1}F_k, \\ y_k &= F_k^T(I + J_{k+1|k+1}Q_k)^{-1}(y_{k+1|k+1} - J_{k+1|k+1}u_k) \end{aligned} \tag{22}$$

And the final solution $X_k^* = \text{argmax}_{X_k} P(X_k|Z)$ is given by merging the estimates obtained at the forward and backward pass as follows for each k, see [26]:

$$X_k^* = x_k + P_k(P_k + J_k^{-1})^{-1}(J_k^{-1}y_k - x_k). \tag{23}$$

However, the given formula implies inverting J_k, which we want to avoid. Therefore, we propose to modify (23) as follows:

$$\begin{aligned} X_k^* &= x_k + P_k(P_k + J_k^{-1})^{-1}J_k^{-1}(y_k - J_k x_k) && (24) \\ &= x_k + (I + P_kJ_k)^{-1}(P_ky_k - P_kJ_kx_k) && (25) \\ &= (I + P_kJ_k)^{-1}(x_k + P_ky_k), && (26) \end{aligned}$$

where the push-through identity [20] shows that $P_k(P_k + J_k^{-1})^{-1}J_k^{-1} = (I + P_kJ_k)^{-1}P_k$. Moreover, the final covariance of X_k^* given Z, $P_{k|N}$ is given by

$$P_{k|N} = (I + P_kJ_k)^{-1}P_k \tag{27}$$

We see the equations above allow performing optimal smoothing without involving at any time the matrix inverses Q^{-1} or P_k^{-1} where P_k denotes the forward covariance matrices, see (17). Indeed, in (22) and (23) each time those matrices are involved in an inversion operation there is a natural regularization term $(I + \cdot)$ involved as well.

C. Summary of the approach

We summarize the approach and the results obtained so far.

- We consider nonlinear system (1) with measurements of the form (3) and (4). In the smoothing approach, one seeks to compute the most likely (entire) trajectory χ of (2) in the light of past measurements. By contrast, the filtering approach is only concerned with computation of the most likely current state X_N.
- The solution to the smoothing problem corresponds to optimization problem (7) with cost function (6). To attack this problem one usually linearizes the cost at current estimate and solves a simplified optimization problem (8) with quadratic cost function (9), that is, a least squares problem. This provides a correction to the current estimate of χ, and then the cost (6) is relinearized at the corrected estimate, yielding another least squares problem which in turn provides a new correction. This is repeated until convergence to the optimum.
- At each optimization step, the least squares solution (8) involves solving the normal equations (12), which may be ill-conditioned when process noise is too low. In the present paper, we address the problem of smoothing in the presence of process noise covariance matrices that may be ill-conditioned or even singular, owing to the use of highly accurate motion sensors.
- In the case of unary observations (3) only, one may associate a linear dynamical system (16) with the linearized cost (9) at each step. The Kalman smoother then provides a solution to the optimization problem (8). By using a slightly modified implementation related to the BIFM of [26], we obtain a solution to (8) without inverting matrices that may be ill-conditioned or singular.
- In the sequel, we seek to adapt the proposed latter solution to the case where measurements of the form (4) are also involved, as typically arises when using vision, notably for SLAM.

IV. Proposed method: the BIFM Kalman smoother with stochastic cloning

Throughout this section, we still consider the linearized optimization problem (8) with cost (9), but where observations may involve pairs of states, to account for measurements of the form (4). As previously done in (16), one may then associate a linear system

$$X_{k+1} = F_kX_k + u_k + w_k, \tag{28}$$

$$Z_k = H_kX_k + V_k \quad \text{(unary observations)} \tag{29}$$

$$Z_k = H_kX_k + H_lX_l + V_k,\ l < k \text{ (pair of states observations)} \tag{30}$$

to the optimization problem, where (30) stems from the linearization of observations of the form (4). As in Section II-A2, Z_k denotes both types of measurements. Building upon the BIFM of the previous section, we will attempt to solve the corresponding optimization problem (i.e., smoothing) without ever inverting matrices that may be ill-conditioned or singular in the presence of low or even null process noise w_k.

A. Stochastic cloning for filtering

Although we are concerned with smoothing, we have seen the Kalman filter is the main component of the forward pass of the BIFM smoother. Standard Kalman filtering for linear systems of the form (28)-(30) has been rendered possible through the stochastic cloning method, introduced in [33] and one of the key components of the MSCKF [34]. The state is cloned at some point in time l and kept in memory to be able to compute an update that will involve it in the future at

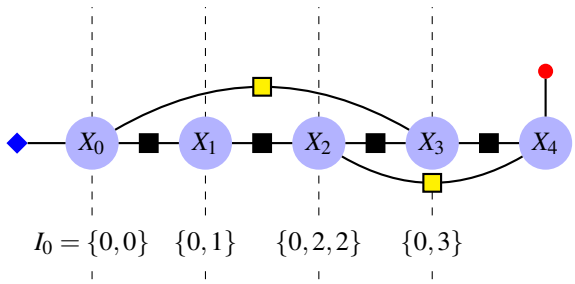


Fig. 2: Stochastic cloning methodology for the forward pass in the factor graph framework. At each time k, state variable X_k is augmented to form the variable $\tilde{X}_k := X_{I_k}$ where I_k consists of current state index k and all the indices of the current and past variables that are to be used later in a relative measurement. In a factor graph representation, variables involved in stochastic cloning are easily visualized: "clones" at time k are the variables related by an orange factor that spans from, to, or "over" the current state.

time k, see (30). The clones are discarded once they are not useful anymore. Note that the variables designated as clones are easily visualized in the factor graph framework, see Figure 2.

Remark 3: Since the focus is put on the linearized problem, the factor graph is fixed, therefore we know in advance which states are linked together, and thus when cloning or discarding. Removing a clone does not delete it from the higher-level estimation process, as does marginalisation for instance.

To incorporate a measurement (30) in the Kalman filter update, one needs the mean and covariance matrix of the vector (X_l, X_k). Thus, denoting by I_k the set containing k and all indices of past states involved in future measurements, we may define the current state at time step k as $\tilde{X}_k = (X_i)_{i\in I_k}$, and we see its size changes over time, see Figure 2.

1) The stochastic cloning pipeline: The goal of Kalman filtering with SC is to compute at each time step k the parameters of the Gaussian density $P((X_i)_{i\in I_k}|Z_0,\dots Z_{k-1})$, and then to update them accounting for observations available at k, that is, compute the parameters of distribution $P((X_i)_{i\in I_k}|Z_0,\dots Z_k)$. This can be done along the lines of the standard Kalman filter, even though the dimension of the state keeps changing over time. The cloning pipeline may be intuitively described as follows:

$$\text{Cloning} \rightarrow \text{Propagation} \rightarrow \text{Update} \rightarrow \text{Clone discarding}$$

Let $\tilde{P}_k$ denote the covariance matrix of the augmented state $\tilde{X}_k = (X_i)_{i\in I_k}$. The SC steps are as follows.

Clone creation (duplication step): The cloning step corresponds to the fact that if the current state X_k is to be used in a future relative measurement, then it must be cloned. This simply consists in duplicating the X_k and concatenating it with the full state $\tilde{X}_k$, which thus now contains the state X_k twice. Indeed, the current state is of the form $\tilde{X}_k$ where I_k contains current index k and index of clones already created and not yet discarded. Duplication of current state then writes:

$$\tilde{X}_k \leftarrow (X_k, \tilde{X}_k) \tag{31}$$

This may be rewritten in terms of matrix computation as follows:

$$\tilde{X}_k \leftarrow C_k \tilde{X}_k, \qquad C_k = \begin{pmatrix} I & & & \\ I & & & \\ & & \ddots & \\ & & & I \end{pmatrix} \tag{32}$$

Since X_k and its copy are fully correlated, the covariance of the extended vector must be replaced with

$$\tilde{P}_k \leftarrow C_k \tilde{P}_k (C_k)^T = \begin{pmatrix} P_k & P_{I_k,k} \\ P_{k,I_k} & \tilde{P}_k \end{pmatrix}, \tag{33}$$

where P_k is the marginal covariance of the current state variable X_k, $\tilde{P}_k$ the marginal covariance of the augmented state $\tilde{X}_k$ before duplication, $P_{I_k,k}$ and P_{k,I_k} the cross-correlations. The computation of P_{k,I_k} is made by copying blocks from $\tilde{P}_k$ and the correlation between X_k and X_k is the identity matrix.

Propagation: Clones remain static during propagation, since the rationale is merely to keep past states in the augmented state. Hence we use a propagation step analogous to (19) but with augmented dynamics, input, and process noise as follows:

$$F_k^p = \begin{pmatrix} F_k & 0 \\ 0 & I \end{pmatrix}, \quad \tilde{u}_k = \begin{pmatrix} u_k \\ 0 \end{pmatrix}, \quad \tilde{w}_k = \begin{pmatrix} G_k w_k \\ 0 \end{pmatrix}. \tag{34}$$

Update: The main purpose of SC is to allow carrying out the updates as in the standard Kalman filter. Indeed, as each relative observation at time k involves a past state variable with index $l < k$ that has been cloned, and thus $l, k \in I_k$, observations (30) may be written using a novel observation matrix $\tilde{H}_k = (H_i)_{i\in I_k}$, defined such that

$$\tilde{H}_k X_{I_k} = \sum_{i\in I_k} H_i X_i. \tag{35}$$

Clone discarding: Once a clone is no longer useful (i.e., it will be involved in no later measurement), it can be discarded. This is done by marginalizing out the considered clone, which reads in terms of matrices

$$\tilde{X}_k \leftarrow D_k \tilde{X}_k, \quad \tilde{P}_k \leftarrow D_k \tilde{P}_k (D_k)^T, \tag{36}$$

where D_k is the identity matrix from which we removed the rows corresponding to states we discard, that is, those in $I_k \setminus I_{k+1}$.

2) SC implementation: Gathering all the steps above, we see the obtained Kalman filter maintains the augmented state $\tilde{X}_k = (X_i)_{i\in I_k}$ and its covariance matrix $\tilde{P}_k$. Formally, $\tilde{X}_k$ follows the state equations

$$\tilde{X}_{k+1} = \tilde{F}_k \tilde{X}_k + \tilde{u}_k + \tilde{w}_k, \tag{37}$$

$$Z_k = \tilde{H}_k \tilde{X}_k + V_k, \tag{38}$$

using the augmented quantities defined in (34), and where the matrix $\tilde{F}_k$ is defined with the help of matrices introduced in (32), (34), and (36), by

$$\tilde{F}_k = F_k^p C_k D_k, \tag{39}$$

(note we suggest to perform the discarding step before duplication to spare the algorithm undesirable computations), and with observations defined by (35) corresponding to (29)-

(30). Formally the obtained system (37)-(38) fits into the standard system form used by the Kalman filter, and in fact the dimension of the state changes does not result in any modification of the Kalman filter method.

B. Proposed "SC-BIFM" (stochastic cloning for smoothing)

The proposed solver consists in applying the BIFM smoother of Section III-B dedicated to system (16) to the augmented system created by stochastic cloning to address systems of the form (28)-(29)-(30). This way, we will achieve robustness to singular prior and process noise covariance matrices.

1) Forward pass with stochastic cloning: The forward pass is akin to Kalman filtering, and the method of Kalman filtering with stochastic cloning was recalled in (19)-(20) and Section IV-A, and put in perspective in the proposed factor graph context. The forward pass consists of the BIFM forward recursion (19)-(20), applied to the augmented system (37)-(38) having augmented state variables $\tilde{X}_k$.

2) Backward stochastic cloning with an information filter: This step consists in applying the backward equations in information form (22) to the augmented system (37)-(38) instead of the standard system dynamics with unary observations (16). As a noticeable difference, we observe that contrary to F_k in (16), matrix $\tilde{F}_k$ is not square. However, only its transpose is involved in the backwards equations and this poses no problem. This is an advantage of having written the backwards recursion in information form: in standard form writing state at time k from state at time $k+1$ would require matrix inversion.

Let $\tilde{y}_k$ be the augmented backward information vector, and $\tilde{J}_k$ the associated information matrix. Let us comment on the steps involved in the backwards pass in the presence of clones.

Clone creation: In the forward pass, the vector state is augmented each time a pose has to be kept for later use, while the covariance matrix is augmented with full correlation as in Eq. (33). In the backward case, the order of the steps is reversed, so the information vector and information matrix are augmented each time a clone was discarded in the forward pass. The process of clone creation and destruction in backwards time strictly mirrors the forward time. However when augmentation occurs in the backward sense new entries of both $\tilde{y}_k$ and $\tilde{J}_k$ are padded with zeros: we create a state with no correlation with the rest of the system. Note indeed that the clone refers to another state variable X_k appearing in the "past", that is, X_l with $l < k$. Thus immediately after creating it the observation linking the current state k to the added state having index l is performed, which will create correlation. Note this interesting fact: in the backward sense the clone does not "know" it is a clone (i.e., is by no means *fully* correlated with any state but only loosely correlated to it via the noisy observations involving the pair of states). Full correlation only appears when it gets discarded, mirroring the forward case where the clone is "informed" it is a clone, through Eq. (33), at its creation.

Clone discarding: In the backward propagation, the counterpart of clone creation is surprisingly simple, since it boils down to applying $(D_k)^T$, see (39) and recall from (36) that D_k is the identity matrix from which we removed the rows corresponding to states we can discard. Assuming the clone lies at position l in the information vector and we are dealing with state at position k, the corresponding operation reads:

$$\begin{aligned} \tilde{y}_k &\leftarrow \tilde{y}_k + \tilde{y}_l \\ \tilde{J}_{k,k} &\leftarrow \tilde{J}_{k,k} + \tilde{J}_{k,l} + \tilde{J}_{l,k} + \tilde{J}_{l,l} \end{aligned} \tag{40}$$

These formulas are directly derived from (22) and (32) reading what happens to the entries of the matrix J and the vector $\tilde{y}$. Indeed, the transpose of C_k is applied, which sums the last two blocks of a vector. Combined with the transpose of the clone discarding matrix D_k, we get the above formulas.

3) Interpretation: Although the backward pass was obtained through "blind" matrix manipulations of the system in reverse time, the following interesting interpretation of what cloning means in reverse time provides insight. Indeed, discarding the clone in backward time is actually a sequence of two actions: informing the clone it is a clone, then killing it (mirroring clone creation and full correlation information (33) in the forward pass). The first action is equivalent to a noise-free observation $Z_k = X_k - X_l$, taking value $Z_k = 0$. To study this operation mathematically, let us associate a non-zero covariance matrix R to observation Z_k. We will then study what happens when $R \to 0$. Since we are in information form, only the components $\tilde{y}_k, \tilde{y}_l$ and the blocks $\tilde{J}_{k,k}, \tilde{J}_{k,l}, \tilde{J}_{l,k}, \tilde{J}_{l,l}$ are affected by this operation. The Kalman update equations then yield:

$$\begin{aligned} \tilde{J}^+ = \begin{pmatrix} \tilde{J}^+_{k,k} & \tilde{J}^+_{k,l} \\ \tilde{J}^+_{l,k} & \tilde{J}^+_{l,l} \end{pmatrix} &= \begin{pmatrix} \tilde{J}_{k,k} & \tilde{J}_{k,l} \\ \tilde{J}_{l,k} & \tilde{J}_{l,l} \end{pmatrix} + \begin{pmatrix} I & -I \end{pmatrix}^T R^{-1} \begin{pmatrix} I & -I \end{pmatrix} \\ &= \begin{pmatrix} \tilde{J}_{k,k} + R^{-1} & \tilde{J}_{k,l} - R^{-1} \\ \tilde{J}_{l,k} - R^{-1} & \tilde{J}_{l,l} + R^{-1} \end{pmatrix} \end{aligned} \tag{41}$$

After this observation, we marginalize out the clone. In information form, the remaining block $\tilde{J}^{++}_{k,k}$ is given by the classical Schur complement formula:

$$\tilde{J}^{++}_{k,k} \leftarrow \tilde{J}^+_{k,k} - \tilde{J}^+_{k,l} \left[\tilde{J}^+_{l,l}\right]^{-1} \tilde{J}^+_{l,k}$$

Replacing the blocks of $\tilde{J}^+$ by their values from Eq. (41), the new block $\tilde{J}^+_{k,k}$ after the sequence " observation + marginalization" reads:

$$\tilde{J}^{++}_{k,k} = \tilde{J}_{k,k} + R^{-1} - \left(\tilde{J}_{k,l} - R^{-1}\right)\left(\tilde{J}_{l,l} + R^{-1}\right)^{-1}\left(\tilde{J}_{l,k} - R^{-1}\right) \tag{42}$$

$$= \tilde{J}_{k,k} + R^{-1} - \left(\tilde{J}_{k,l}R - I\right)\left(\tilde{J}_{l,l}R + I\right)^{-1}\left(\tilde{J}_{l,k} - R^{-1}\right) \tag{43}$$

$$= \tilde{J}_{k,k} + R^{-1} - \left(\tilde{J}_{k,l}R - I\right)\left(I - \tilde{J}_{l,l}R + O(R^2)\right)\left(\tilde{J}_{l,k} - R^{-1}\right), \tag{44}$$

where, after factorising R^{-1} in $\left(\tilde{J}_{ll} + R^{-1}\right)^{-1}$, the first-order expansion $(I+\varepsilon)^{-1} = I - \varepsilon + O(\varepsilon^2)$ was carried out. Developing the parentheses, we see a $-R^{-1}$ term appears, canceling the R^{-1} term (second term in the right hand side), and we end up with:

$$\tilde{J}^{++}_{k,k} = \tilde{J}_{k,k} + \tilde{J}_{k,l} + \tilde{J}_{l,k} + \tilde{J}_{l,l} + O(R^2)$$

Algorithm 1: SC-BIFM

Input: $P_0, (u_i)_i, (Q_i)_i, (Z_k)_k, (R_k)_k$;

Forward pass

1 Set $\tilde{X}_0 = Z_0$, $\tilde{P}_0 = P_0$;

For $k < N$ **do**

2 Compute $\tilde{X}_{k+1}$ and $\tilde{P}_{k+1}$, based on (37), (38), using the Kalman equations (19), (20);

Backward pass

3 Set $\tilde{y}_N = 0$, $\tilde{J}_N = 0$, $k = N$;

For $k > 0$ **do**

4 Compute $\tilde{y}_{k-1}$ and $\tilde{J}_{k-1}$, based on (37), (38) using the information form equations (22), (21) ;

For $k \leq N$**, do**

5 Compute the augmented solution $\tilde{X}^*_{I_k}$ based on (26) and extract only the state variables X^*_k ;

Output: $\chi = (\tilde{X}^*_k)_{k \leq N}$ which is the exact solution to the linearised optimization problem (9);

And finally we can make R tend to zero (i.e., precision of the virtual measurement to infinity) to recover (40). Interestingly, the action of "informing two states they are clones", easily encoded in the covariance form, cannot be made in the information form, due to the infinite R^{-1} terms. On the other hand the sequence ""informing two states they are clones, and then keeping only one" works out beautifully in the information matrix form.

4) Final fusion: The final fusion is carried out for each step as in standard BIFM, according to (26). The full pipeline is summarised in Algorithm 1.

Remark 4: Alternatively, at step 5 of Algorithm 1 one could first extract simple state variables X_k from augmented ones $\tilde{X}_k = X_{I_k}$, and then perform fusion (26) with smaller matrices to save computation time. However the extraction is a marginalization, which has to be performed on the information variables $\tilde{y}$, $\tilde{J}$, which is computationally costly.

Note that from the final update we may also get the posterior covariances of the related variables. However, this only yields covariances of variables which are neighbors in the measurement graph, contrary to more general methods [22]. To this respect, note that an alternative to SC-BIFM, which does not try to maintain sparsity, is to simply not discard the created clones. Then, at the end of the forward pass, the filter outputs $P(X_0, \ldots, X_n | Z_0, \ldots, Z_n)$, that is, the solution we pursue.

V. Numerical illustration of the methodology

We illustrate numerically the behavior of each method using a linear simplified navigation example. Consider a body equipped with a biased accelerometer moving along a horizontal line. By letting $p(t)$ denote its position, $v(t)$ its velocity, and b a static accelerometer bias, the noiseless dynamical motion equations read

$$\dot{p} = v, \;\; \dot{v} = (u + b), \dot{b} = 0$$

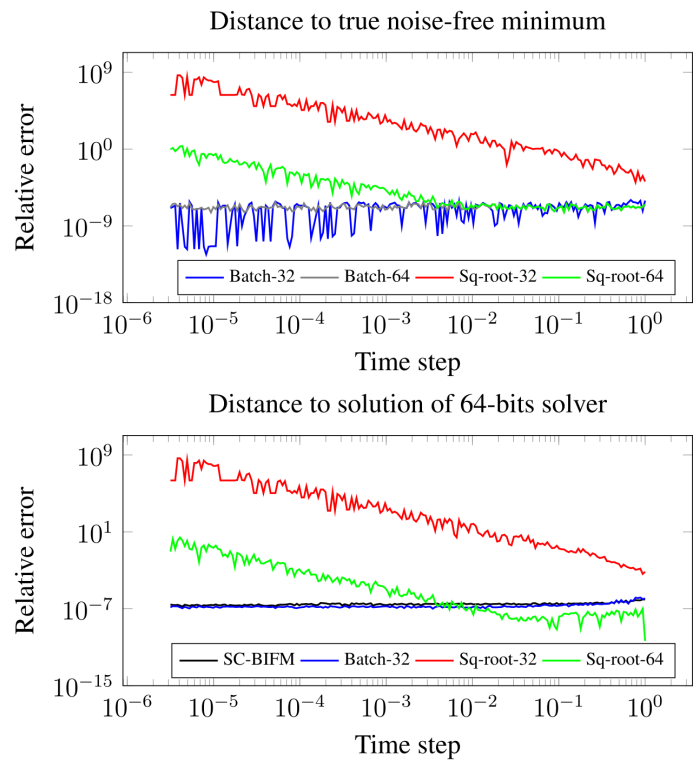


Fig. 3: Top plot: distance between the solutions of various solvers and the true minimum of the simple numerical example of Section V in the process noise free case. Bottom plot: average Monte-Carlo distance to the solution of the 64-bits robust batch solver (which systematically finds the optimum) for various solvers in the presence of process noise. In both cases performances are plotted against the discretization time step dt in (46). Standard solvers based on the information form (Sq-root-32 and Sq-root-64) degrade rapidly when dt is taken small, whereas the ones we propose (Batch and SC-BIFM) do not degrade, even with single precision (32 bits).

with u the measurement reported by the accerelorometer. We suppose morevoer that at discrete time instants, using for instance vision, the system is able to measure relative displacement with respect to a past position. In discrete time, adding some process and measurement noise, and following the structure of the graph of Figure 1, the state $X = (b, v, p)$ is approximately governed by

$$X_{k+1} = \begin{pmatrix} 1 & & \\ -dt & 1 & \\ & dt & 1 \end{pmatrix} X_k + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} u_k + w_k \tag{45}$$

$$Z_3 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} (X_3 - X_0) + V_k \tag{46}$$

$$Z_4 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} (X_4 - X_2) + V_k \tag{47}$$

This model is a simplified version of the system dynamics used in inertial navigation. Usually, position integration with respect to the speed is considered exact on a single time step, however IMU outputs are given at a much higher rate than the observations, and are therefore preintegrated between two states [18]. This leads to a propagation factor with full-rank (although low eigenvalues) covariance matrix Q. We take the

length of the trajectory to be $N = 4$. Observation noise V_k has variance σ_z. The process noise w_k has diagonal covariance $\mathrm{diag}(\sqrt{dt}\sigma_b, \sqrt{dt}\sigma_{acc}, \sqrt{dt}\sigma_{int})$, which represent the bias random walk, the accelerometer and the integration uncertainty respectively. Their magnitude reflects typical characteristics of high-grade inertial sensors:

$$\sigma_b = 1e-3,\ \sigma_{acc} = 1e-2,\ \sigma_{int} = 1e-3, \tag{48}$$

with initial standard deviations and observation noise

$$\sigma_b^0 = 1e-2,\ \sigma_v^0 = 1,\ \sigma_p^0 = 1, \sigma_z = 0.1. \tag{49}$$

We compare the following solvers:

- The standard square-root information solvers based on Cholesky (or QR) factorisation of Section II-E, with precision of 64 (Sq-Root-64) and 32 bits (Sq-Root);
- The robust batch solver ("Batch solver") of Section II-H, with precision of 64 (Batch-64) and 32 bits (Batch 32);
- The proposed SC-BIFM solver in 32 bits (only).

A. Ideal case numerical experiment

First, we study the impact of conditioning on the solvers in the noise free case, that is where the ground truth is described by (45), (46) with $w_k = 0$ and $V_k = 0$. The state is initialised at the ground-truth, so that we can provide the true minimum as ground truth, by simply integrating (45). The estimation here is made relative to the initial position p_0, which is naturally considered fixed and is removed from the state. The effect of the time step dt on the numerical behavior of the various solvers is displayed on Figure 3, top plot. Indeed, dt is tightly linked with the system's conditioning, since the propagation covariance varies with its square-root.

We can see that, as expected, the solvers based on information matrix degenerate as the time step becomes too small, while the others maintain a stable relative error, in both 32 and 64 bits. SC-BIFM is not displayed here, as it exactly follows the true state owing to perfect initialization in the forward pass.

B. Numerical experiment with low noise

Monte-Carlo simulations were run with noise turned on. Since the 64 bits batch robust solver actually managed to find the true solution of the linear least-squares problem in all cases, it was taken as the reference for the comparison of the remaning solvers. 200 sets of measurements were randomly generated with noises (48)-(49). The distance between the solution of each solver and the ground-truth are computed at each run, and the average distances are shown in Figure 3, bottom plot. In the noisy case, we also see that the standard solvers perform equivalently or slightly better than the robust ones at large time steps, but degrade rapidly for smaller ones . These results clearly indicate that one should be careful when designing least-squares-based estimators for accurate sensors.

Remark 5: The degeneracy of the square-root filters comes from the fact that they are based on QR factorisation. Using an SVD to invert the system proves to be much more robust. However, it does not maintain sparsity and is therefore not desirable in the context of factor graph navigation and SLAM.

Fig. 4: Car used in the experiments.

VI. Real world experiment

This section deals with a real high precision inertial-LiDAR odometry problem in the context of autonomous cars. We conducted our own experiments, using the experimental car of Safran shown in Figure 4. In our approach, 3D LiDAR scans between keyframes were first processed to obtain relative transformations using scan matching algorithms. This imposes observations between pairs of state variables at different times of the form (4), which are then fused with inertial measurements. Combining pose from relative scans with odometry through graph optimization has been pursued in various works, see for example [31]. However, combining relative scans with IMU has been less studied, and to our knowledge pose graph SLAM using LiDAR and high grade military (or "tactical") precision IMUs has never been done before. This is the main originality of the following experiments, along with a suitable optimization approach to handle such levels of precisions that lead to ill-conditioned normal equations.

A. Considered models

We use the standard IMU preintegration framework, see [18] to define our propagation factors. The LiDAR scans are aligned independently to obtain relative poses. Let $X_k = (R_k, v_k, p_k, b_k^\omega, b_k^a)$ represent the state variables: $R_k, v_k, p_k, b_k^\omega, b_k^a$ are the orientation, velocity, position and biases. This leads to the following dynamics, see e.g., [18]:

$$\begin{aligned} R_{k+1} &= R_k \exp_{SO(3)}(dt\ \omega_k - b_k^\omega + w_k^\omega) \\ v_{k+1} &= v_k + dt\ (R_k(a_k - b_k^a + w_k^a) + g) \\ p_{k+1} &= p_k + dt\ v_k \\ b_{k+1}^\omega &= b_k^\omega + w_k^{b\omega} \\ b_{k+1}^a &= b_k^a + w_k^{b_a}, \end{aligned} \tag{50}$$

where $w_k^\omega, w_k^a, w_k^{b\omega}, w_k^{b_a}$ are white noises, associated to the isotropic covariances of standard deviations $\sigma_\omega, \sigma_a, \sigma_{b^\omega}, \sigma_{b^a}$ respectively.

B. Experimental setup

All experiments were performed using the car shown in Figure 4, which is equipped with a high-grade IMU from

the company Safran Electronics and Defense and a Velodyne VLP32C. The car drove along an $11km$ long loop in a suburban area, including medium-speed ($90km/h$) portions and roundabouts. The IMU provided data at 100 Hz, for which we set, according to the specification given by the manufacturer:

$$\begin{aligned}
P_0 &= \mathrm{diag}((\sigma_R^0\ I_2, 0), \sigma_v^0\ I, 0, \sigma_{b^\omega}^0\ I, \sigma_{b^a}^0\ I),\\
\sigma_R^0 &= 1\ ^\circ,\ \sigma_v^0 = 100\ m/s,\ \sigma_{b^\omega}^0 = 0.01\ ^\circ/s,\\
\sigma_{b^a}^0 &= 0.05\ m/s^2, \sigma_\omega = 0.1\ ^\circ/h\\
\sigma_a &= 0.0015\ m/s^2\ (0.15mg),\ \sigma_{b^\omega} = 2.5e-5\ ^\circ/s/\sqrt{s},\\
\sigma_{b^a} &= 8e-4\ m/s^{-2}/\sqrt{s}.
\end{aligned}$$

From the acquired 3D laser scans, relative transforms were obtained at 4 Hz. Scan matching algorithms such as the well-known iterative closest point (ICP) [3], or more recent methods [46], [13], return relative orientation and translation. As the gyro only drifts of 0.1 degree in 1 hour, the level of uncertainty associated to the relative rotation between LiDAR scans computed by the ICP is much higher than the gyro's uncertainty. On the other hand, we estimated relative translations between scans computed using the ICP have an accuracy of order 10 cm (and the measurement covariance matrix R is set accordingly), which is more precise than the relative displacement obtained via a double integration of the accelerometer. As a result we use relative translations as an observation between pairs of states (4).

C. Implementation

A standard smoothing on-manifold approach was used to carry out the estimation to cope with the nonlinear structure of rotations. A 5 seconds sliding window approach was adopted, in which states are added each time a relative translation measurement is received. Once the number of estimated states exceeds the window's size, the oldest state is marginalised out. The Gauss-Newton strategy described in Section II was used, with one descent iteration carried out each time a measurement was acquired. Different solvers were implemented for comparison purposes. To focus on them, the rest of the pipeline was fully treated in double precision. All algorithms were implemented in Python, based on "Numpy" and "Scipy" built-in functions. In the following we detail the specifics of the implementation of the IMU preintegration and the solvers.

Computing the uncertainties of the prior and the IMU preintegration: The preintegration of IMU increments was done following [18]. The associated uncertainty, however, was computed by propagating the square-root of the associated covariance [42]. Likewise, the uncertainty of the prior was computed in square-root form. This was needed to avoid some numerical issues, and was useful for feeding the Square-Root solvers with the most accurate uncertainty estimation.

SC-BIFM: It was implemented according Algorithm 1. The covariance matrices P_0 and Q were retrieved by simply squaring the triangular matrices stored for the Prior and the IMU.

Square-Root Smoothing: Square-Root smoothing was achieved using the QR decomposition of $\tilde{A}$, based on its "Numpy" implementation. Then the resulting linear systems were solved thanks to the "Scipy" package.

Robust Batch Approach: As for SC-BIFM, the Batch solver was fed with full covariance matrices (i.e., not square roots). The solution was computed using the "Scipy.sparse" package, and especially its "spsolve" method to invert the ensuing square linear systems.

D. Experimental comparison of linear solvers

Three linear least squares solvers are compared, in their 32- and 64-bits floating point formats:

- The standard square-root information solvers based on Cholesky (or QR) factorisation of Section II-E, with precision of 64 (Sq-Root-64) and 32 bits (Sq-Root-32);
- The robust batch solver (Batch solver) based on reformulation of Section II-H, with precision of 64 (Batch-64) and 32 bits (Batch 32);
- The proposed SC-BIFM solver in 64 and 32 bits.

As noticed in the simple numerical example of Section V, the best method in terms of accuracy (but not in terms of complexity) is the Batch formulation of Section II-H which solves the linear least squares while avoiding ill-conditioned related issues. As a result, to provide all solvers meaningful linearization points, we performed fusion betwen IMU and relative pose measurements based on scan matching using the Batch solver with 64 bits based double precision.

Each time step where a GN iteration was carried out, the condition number of the weighted jacobian $\tilde{A} = \Sigma^{-1/2}A$ associated to the full normal equations (12), and all the solvers were applied to the corresponding linear least squares system. Figure 5, shows the evolution of the condition number in log-scale, and the distance between the solution of each of the single precision solvers with respect to that of the Batch-64, which is considered herein as ground truth.

Accuracy: The methods rank as follows. The Batch approach we proposed in Section II-H runs first, then the proposed SC-BIFM, and finally the square root resolution of full normal equations. The fact that Batch beats SC-BIFM may be due to our taking advantage of the existing libraries to solve the linear systems that involve A_1^{-1}, whereas SC-BIFM uses no such library owing to its different nature and implementation based on Kalman smoothing, and thus had to be coded entirely from scratch.

Execution time: Average computation times of the various solvers are displayed in Table I are competitive with the Sq-Root, in both precision formats. Batch solvers are slightly slower, which was expected because of the A_1^{-1} term to be computed. This is on the other hand encouraging for SC-BIFM, which is based on an academic code developed from scratch and thus does not yet benefit from the same level of code optimisation as the other methods that use bricks such as Numpy built-in QR decomposition. Theoretically, our code runs in $O(m^3n)$, where n is the number of variables and m the dimension of the largest X_{I_k} related to the number of clones. It is difficult to go further into complexity analysis owing to the clones, but we believe its complexity is closely related to the complexity of sparse Cholesky or QR decompositions, which varies with the filling of the R factor.

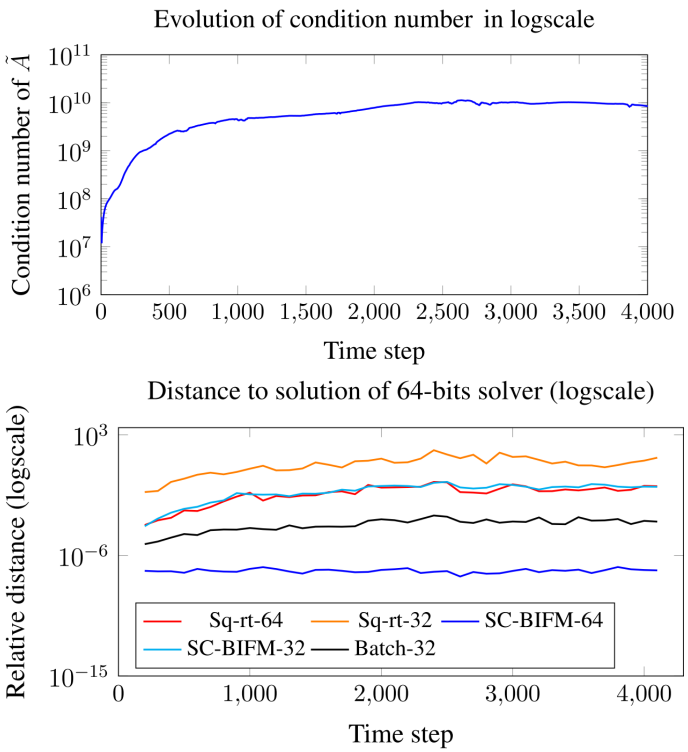


Fig. 5: Top : Evolution of the condition number of $\tilde{A}$ during the trajectory, in log scale; Bottom : Distance to Batch 64 considered herein as ground truth, in log scale. A moving average on 100 steps was performed to improve readability. There is a clear link between the increase of ill-conditioning and the accuracy degradation of the single precision solvers. However, the proposed SC-BIFM systematically beats its square-root information counterpart described in Section II-E. Notably it is remarkable SC-BIFM achieves comparable results with 32 bits single-precision as square root with 64 bits double-precision implementation.

Solver	Average computation time (s)
Sq-Root 64	0.021
Sq-Root 32	0.010
Batch 64	0.029
Batch 32	0.024
SC-BIFM 64	0.020
SC-BIFM 32	0.016

TABLE I: Average computation times for the inversion of the linear systems over trajectory. Computations were made on a standard laptop with Intel i5-5300 2.3 GHz CPU

Solver	Maximum distance to Batch-64 (m)
Sq-Root 64	0.09
Sq-Root 32	∞
Batch 64	0
Batch 32	0.01
SC-BIFM 64	$7e{-}4$
SC-BIFM 32	0.1

TABLE II: Maximum distance to the Batch-64 trajectory for the various solvers.

E. Experimental comparison of corresponding localization algorithms

Following the full pipeline of iterative linearization procedure of factor graph based SLAM (or navigation) recapped in Section II, we computed state variable estimates for the nonlinear system (50) using the various solvers.

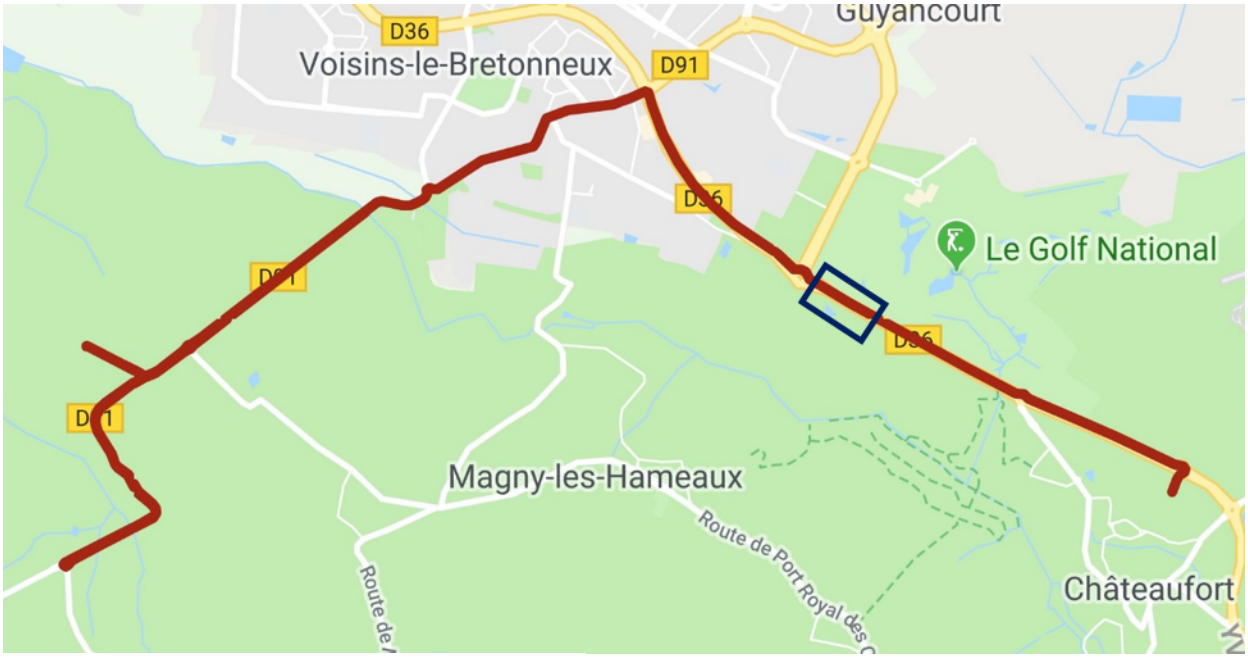
Table II displays the maximum distance to Batch 64 estimate (considered as optimal) in terms of position discrepancy of the car. The results confirm what could be anticipated from the linear solvers comparison above. Moreover, using single-precision Sq-root-32, the estimation could not even be carried out until the end, as it diverged after about an eighth of the trajectory, see Figure 6. SC-BIFM-64 is as expected the closest to Batch-64, followed by Batch-32, while Sq-root-64 and SC-BIFM-32 show similar behaviors. The latter feature is remarkable, though, as it shows he proposed algorithm SC-BIFM may achieve good results in single precision on a real application.

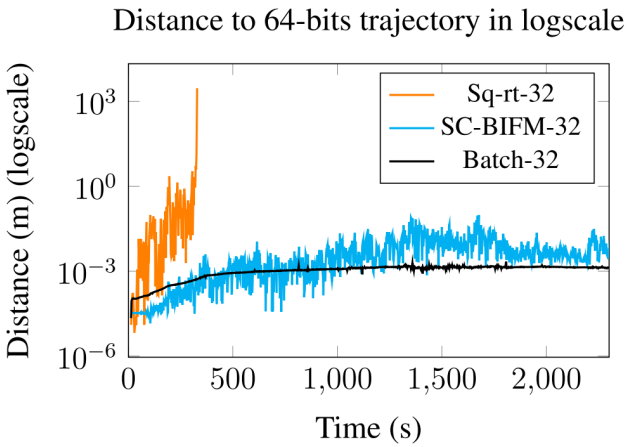
F. Conclusions regarding real experiments

The first merit of the conducted experiments is to prove that the problem of ill conditionned information matrix for pose graph SLAM or Kalman smoothing for navigation may arise when using highly precise inertial sensors (such as cost effective high precision IMU from Safran Electronics and Defense). Given the progress made in the field of inertial sensors over the past decades, we can anticipate performance will keep increasing, and cost decreasing. We showed the resulting problems can even lead to divergence of the localization estimate based on the standard Cholesky resolution of linear least squares (Sq-Root 32) when implemented with 32 bits single-precision. This is an important point, as most industrial grade inertial navigation embedded systems use 32-bit precision. In this respect, we see the two solutions we proposed in this paper are satisfactory and achieve good performance in single precision. Moreover, the proposed novel SC-BIFM method based on Kalman smoothing with stochastic cloning seems promising, as its complexity is reasonably low since only matrices of a limited size need to be inverted, as opposed to the proposed Batch solver.

VII. Conclusion

In this paper we highlight a shortcoming of the standard solvers used to invert the linear least-squares problems appearing in smoothing methods for sensor fusion, and especially





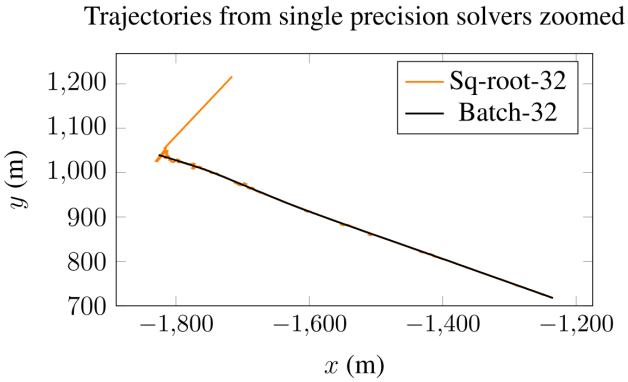


Fig. 6: Top : 11 km long trajectory followed by the car, taken from Google Maps. The bottom right image displays a zoom on the zone inside the black box, and shows the trajectories estimated by Sq-root-32 and Batch-32. We see Sq-root-32 diverges whereas Batch-32 properly follows the true trajectory. Both proposed single precision solvers, SC-BIFM-32 and Batch-32, managed to stay stable during the whole experiment, as shown in the bottom left picture. More generally, the deviations observed are consistent with the results reported in Table II.

inertial navigation. It was shown that these methods are not adapted to overly low propagation noise, and can degenerate, for instance in single precision. We build on the fact that some formulations of Kalman smoothers avoid this numerical issue and propose a novel sparsity-maintaining solver for a large class of systems, using stochastic cloning. It exhibits much higher robustness than standard solvers in the face of the mentioned numerical difficulties.

Research to achieve numerical efficiency comparable to the one of existing solvers is still needed, as they benefit from decades of numerical algorithms, along with major recent breakthroughs from the SLAM community. Another route for future research is to come up with an incremental formulation of SC-BIFM. Indeed, the forward part is easily continued when a new measurement is available, but the backward information filter seems to have to be recomputed from scratch each time. We anticipate that there might be links with the Bayes tree used in iSAM2 [23], for instance, which could help resolving this issue.

VIII. Acknowledgements

This work is supported by the company Safran through the CIFRE convention 2016/1444.

References

[1] A. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] E. Aghapour and J. A. Farrell. Performance specified state estimation with minimum risk. In *2018 Annual American Control Conference (ACC)*, pages 1114–1119, June 2018.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[4] Gerald J Bierman. *Factorization Methods for Discrete Sequential Estimation*, volume 128. Academic Press, 1977.

[5] M. Brossard, A. Barrau, and S. Bonnabel. Exploiting symmetries to design ekfs with consistency properties for navigation and slam. *IEEE Sensors Journal*, 19(4):1572–1579, Feb 2019.

[6] M. Brossard, A. Barrau, and S. Bonnabel. Rins-w: Robust inertial navigation system on wheels. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2068–2075, 2019.

[7] M. Brossard and S. Bonnabel. Learning wheel odometry and imu errors for localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 291–297, May 2019.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[9] Paul Chauchat, Axel Barrau, and Silvere Bonnabel. Invariant Smoothing on Lie Groups. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018*, pages 1703–1710, Madrid, Spain, October 2018.

[10] Han-Pang Chiu, Stephen Williams, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. Robust vision-aided navigation using sliding-window factor graphs. In *2013 IEEE International Conference on Robotics and Automation*, pages 46–53. IEEE, 2013.

[11] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[12] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[13] J. Deschaud. Imls-slam: Scan-to-model matching based on 3d data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485, May 2018.

[14] Gamini Dissanayake, Paul Newman, Hugh F. Durrant-Whyte, Steven Clark, and M. Csobra. A solution to the simultaneous localisation and mapping (slam) problem. *IEEE Trans. Robot. Automat.*, 17:229–241, 2001.

[15] Hugh F. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[16] Ryan M. Eustice, Hanumant Singh, and John J. Leonard. Exactly sparse delayed-state filters for view-based slam. *Trans. Rob.*, 22(6):1100–1114, December 2006.

[17] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2008.

[18] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, Feb 2017.

[19] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, pages 4376–4384, 2016.

[20] Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

[21] Simon J Julier and Jeffrey K Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243. IEEE, 2001.

[22] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and autonomous systems*, 57(12):1198–1210, 2009.

[23] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J. Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31:217–236, 2012.

[24] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, May 2011.

[25] Haomin Liu, Mingyu Chen, Guofeng Zhang, Hujun Bao, and Yingze Bao. Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] H. Loeliger, L. Bruderer, H. Malmberg, F. Wadehn, and N. Zalmai. On sparsity by nuv-em, gaussian message passing, and kalman smoothing. In *2016 Information Theory and Applications Workshop (ITA)*, pages 1–10, Jan 2016.

[27] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Robert E. Mahony and Tarek Hamel. A geometric nonlinear observer for simultaneous localisation and mapping. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, pages 2408–2415, 2017.

[29] Philippe Martin and Erwan Salaün. Design and implementation of a low-cost observer-based attitude and heading reference system. *Control Engineering Practice*, 18(7):712–722, February 2010.

[30] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[31] Ellon Mendes, Pierrick Koch, and Simon Lacroix. Icp-based pose-graph slam. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 195–200. IEEE, 2016.

[32] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[33] A. I. Mourikis, S. I. Roumeliotis, and J. W. Burdick. Sc-kf mobile robot localization: A stochastic cloning kalman filter for processing relative-state measurements. *IEEE Transactions on Robotics*, 23(4):717–730, Aug 2007.

[34] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.

[35] T. Pfeifer and P. Protzel. Robust sensor fusion with self-tuning mixture models. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3678–3685, Oct 2018.

[36] T. Pfeifer and P. Protzel. Expectation-maximization for adaptive mixture models in graph optimization. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3151–3157, 2019.

[37] F. Rahman, E. Aghapour, and J. A. Farrell. Outlier accommodation by risk-averse performance-specified nonlinear state estimation: Gnss aided ins. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5922–5927, Dec 2018.

[38] Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[39] D. M. Rosen, M. Kaess, and J. J. Leonard. Robust incremental online inference over sparse factor graphs: Beyond the gaussian case. In *2013 IEEE International Conference on Robotics and Automation*, pages 1025–1032, May 2013.

[40] D. M. Rosen, M. Kaess, and J. J. Leonard. Rise: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Transactions on Robotics*, 30(5):1091–1108, Oct 2014.

[41] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer, 2016.

[42] F. Wadehn, L. Bruderer, V. Sahdeva, and H. Loeliger. New square-root and diagonalized kalman smoothers. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1282–1290, Sep. 2016.

[43] M. Wang and A. Tayebi. Geometric nonlinear observer design for slam on a matrix lie group. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1488–1493, Dec 2018.

[44] Miaomiao Wang and Abdelhamid Tayebi. A globally exponentially stable nonlinear hybrid observer for 3d inertial navigation. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1367–1372, 2018.

[45] Kejian Wu, Ahmed Ahmed, Georgios A Georgiou, and Stergios I Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*, volume 2, 2015.

[46] Ji Zhang and Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference (RSS)*, Berkeley, CA, July 2014.

[47] Sheng Zhao, Yiming Chen, Haiyu Zhang, and Jay A. Farrell. Differential gps aided inertial navigation: a contemplative realtime approach. *IFAC Proceedings Volumes*, 47(3):8959 – 8964, 2014. 19th IFAC World Congress.