2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Consistent Map-based 3D Localization on Mobile Devices

Ryan C. DuToit[†], Joel A. Hesch[‡], Esha D. Nerurkar[‡], and Stergios I. Roumeliotis[†]

***Abstract*— In this paper, we seek to provide *consistent*, real-time 3D localization capabilities to mobile devices navigating within previously mapped areas. To this end, we introduce the Cholesky-Schmidt-Kalman filter (C-SKF), which explicitly considers the uncertainty of the prior map, by employing the *sparse* Cholesky factor of the map's Hessian, instead of its dense covariance–as is the case for the Schmidt-Kalman filter. By doing so, the C-SKF has memory requirements typically linear in the size of the map, as opposed to quadratic for storing the map's covariance. Moreover, and in order to bound the processing needs of the C-SKF (between linear and quadratic in the size of the map), we introduce two relaxations of the C-SKF algorithm: (i) The sC-SKF, which operates on the Cholesky factors of independent sub-maps resulting from dividing the map into overlapping segments. (ii) We formulate an efficient method for sparsifying the Cholesky factor by selecting and processing a subset of loop-closure measurements based on their temporal distribution. Lastly, we assess the processing and memory requirements of the proposed algorithms, and compare their positioning accuracy against other inconsistent map-based localization approaches that employ measurement-noise-covariance inflation to compensate for the map's uncertainty.**

I. Introduction

In many applications (e.g., surveillance, manufacturing, virtual and augmented reality), robots or people need to accurately localize within a frequently-visited indoor space. In such cases, the accuracy and efficiency of localization can be significantly improved by using a map of the area. In the context of 3D visual-inertial localization, maps computed beforehand[1] have been employed by localization algorithms, such as the multi-state constraint Kalman filter (MSCKF) in [3] and [4], and parallel tracking and mapping (PTAM) in [5] and [6], to improve positioning accuracy based on visual observations of mapped features. These methods achieve real-time performance at the expense of consistency.[2] Specifically, [5] and [6] are inconsistent not only because of the assumption of a perfect map but also due to the approximations invoked by the optimization algorithm used for localization; thus, they cannot provide a reliable measure of their positioning uncertainty. On the other hand, [3], which also assumes that the map is perfect and [4], which ignores the correlations between the estimated state and the map, inflate the camera measurement's noise covariance so as to reduce the effect of inconsistency: overly confident and often unreliable estimates. They provide, however, no guarantees that such approximations will yield consistent estimates. Moreover, inflating the measurement noise does not alleviate the inconsistency that arises from ignoring cross-correlations between the estimated state and the map.

An alternative, approximate method, which explicitly accounts for the map's uncertainty and its correlations with the estimated state, is the Schmidt-Kalman filter (SKF) [8], [9]. The SKF has processing requirements *linear* in the map's size, as it only needs to update the device's state, covariance, and cross-correlation with the map. Although the map's state and covariance need not be updated, storing its covariance has cost *quadratic* in the number of features. This has been the main drawback of the SKF, as well as of its variants applied to simultaneous localization and mapping (SLAM) (e.g., [10], [11], [12]), which has restricted its use to small areas.

To overcome this limitation, in this work, we introduce the Cholesky (C)-SKF which has, typically, memory requirements *linear* in the map's size, while providing the same *consistency* guarantees as the SKF.[3] The key insight behind our approach is that most current methods employed for constructing large-scale maps, such as batch least squares (BLS), compute and use the Cholesky factor of the problem's Hessian, which is sparse (the number of non-zero elements typically increase linearly with the map's size) instead of the dense covariance matrix [14]. The C-SKF's memory savings, however, come at the cost of increased processing requirements. Specifically, depending on the structure of the Cholesky factor, the run time increases between linear and quadratic in the map's size, which quickly becomes prohibitive for mobile devices. To address this limitation, we introduce two *consistent* relaxations of the C-SKF with the objective of reducing the map's size and sparsifying the map's Cholesky factor:

Sub-mapping: In the context of SLAM, a popular approach for reducing the processing cost of a large map is to partition it into independent sub-maps (e.g., [15], [16]). Motivated by sub-map SLAM, in this work, we introduce the sub-map (s)C-SKF, which trades localization accuracy for processing speed by operating on the Cholesky factors of the partitioned Hessians resulting from dividing the original map into independent sub-maps. The sub-maps used throughout this work are generated from the method of [17] due to its inherent ability

[†] R. C. Dutoit and S. I. Roumeliotis are with the Department of Computer Science, University of Minnesota {dutoit,stergios}@cs.umn.edu

[‡] J. A. Hesch and E. D. Nerurkar are with Google Inc. {joelhesch, eshanerurkar}@google.com

This work was supported by Google, Project Tango.

[1] Besides batch least squares (BLS), pose-graphs [1] and PTAM [2] have also been used for reducing the processing cost of map building. Since such approximations yield *inconsistent* maps, we do not consider them further in the context of this work.

[2] As in [7], we define a consistent estimator as one who's estimation errors are zero mean and have covariance matrix smaller or equal to the one calculated by the filter. For the purposes of this work, we focus on the covariance requirement.

[3] These consistency properties refer to the SKF's ability to account for map-uncertainty and device-map cross correlation. Inconsistency can also arise from linearization errors; this case is covered in-depth in [7] and the approach introduced in [13] is employed in this work.

to consistently relax the information available from each sub-map. Note though, that other sub-mapping algorithms could be used.

Map sparsification: When creating a map with many loop closures or hovering over the same scene for extended periods of time, its Hessian (and hence Cholesky factor) can become prohibitively dense, causing the C-SKF to run out of computational resources and/or memory. To address this issue, we seek an efficient method for sparsifying the map's Hessian (and thus Cholesky factor) while maintaining a high-accuracy map. To do so, we first find the optimal (in the BLS sense) estimate for the map and then, *after solving*, we consistently sparsify the map's Hessian, using the BLS estimate as the map's lineraization point. This is in contrast to existing map-sparsification techniques, in which the map is sparsified *while solving* the system, yielding a sub-optimal (in the BLS sense) solution (but achieving faster solve times).

Specifically, several methods focus on limiting the map's size by marginalizing states, then sparsifying dense constraints introduced during marginalization. These include those which rely on variants of Chow-Liu trees to provide either a consistent [18] or inconsistent ([19], [20], [21]) sparse approximation of the information in the marginalization clique; and [22], which altogether drops some of the marginalization-induced constraints. Others optimize over the map's Hessian ([23], [24] with an ℓ_1 regularization term and [25] which constrains pre-selected Hessian elements to be zero) to enforce sparsity with consistency constraints, while minimizing the difference from the original system. Finally, [26] provides a convex-optimization based method to select and discard measurements from the system with the goal of minimizing the resulting solution's deviation from the solution found when employing all measurements.

While the consistent methods listed above ([18], [22], [23], [25], [26], [24]) could be modified to sparsify the map *after solving*, we instead introduce an alternative approach that does not marginalize states (as in [18], [22], and [24]) or have restrictive computational cost for large or highly-connected maps (which is the case for [23], [25], and [26]). The proposed heuristic allows us to reduce the Cholesky fill-ins by efficiently selecting only a subset of the available loop-closure measurements for Cholesky construction.

Sub-mapping and sparsification allow us to employ maps of large areas for *consistent* localization on resource-constrained mobile devices, such as cell phones and tablets in real time.

In summary, the main contributions of this paper are:

- We introduce the Cholesky-Schmidt-Kalman filter (C-SKF), which employs the Hessian's Cholesky factor to compactly represent the map's uncertainty, and efficiently compute *consistent* map-based updates.
- We present the sub-map (s)C-SKF, a relaxation of the C-SKF, which employs multiple, independent sub-maps of the area of interest to support real-time consistent map-based localization on mobile devices.
- We employ a sparsified Cholesky factor of the (sub-)map's Hessian (without modifying the map estimate) so as to reduce the computational requirements of the (s)C-SKF when processing observations to (sub-)maps with excessive loop-closures and/or prolonged hovering.
- We validate the accuracy and consistency of our methods using visual and inertial measurements from mobile devices against VICON ground truth.

In what follows, we provide an overview of our map-based localization system (Sect. II) and then present the system state and measurement models (Sect. III). In Sect. IV, we describe the limitations of the SKF when applied to map-based localization, and then introduce the C-SKF and the sC-SKF. Our method for sparsifying pre-existing (sub-)maps is presented in Sect. V. Lastly, we experimentally validate the proposed algorithms in Sect. VI and provide concluding remarks with a discussion on future work in Sect. VII.

II. Map-based Localization Algorithm Overview

Our objective is to design a *consistent* estimator for computing the 3D pose (position and attitude) of a mobile device in real time, using inertial and visual measurements, as well as a prior map of the area of operation. To do so, we require the following information from the (offline) mapping process:

- The (sub-)map's estimated state (i.e., point features and camera poses w.r.t. which they are expressed).[4]
- The Cholesky factor of the (sub-)map's Hessian.
- Binary feature descriptors (in our implementation FREAKs [27]) and the vectors of their corresponding images indexed by a vocabulary tree (VT) [28].

The first two are necessary for representing the map and its uncertainty, respectively, while the last is used for recognizing mapped features. Given this information, we hereafter provide an overview of our online map-based localization system:

At the core of our estimator is the MSCKF ([3], [29]) which processes inertial measurements for propagating the device's state and covariance estimates. When feature tracks (e.g., Harris corners [30] tracked by KLT [31]) become available, the proposed (s)C-SKF adaptation of the MSCKF consistently updates the device's state-covariance and correlations with the map, but *not* the map itself (Sect. IV-C). Similarly, each time the 2D-to-3D feature-matching pipeline (detailed in [32]) identifies correspondences between the features extracted in the current image and those found in the map, the (s)C-SKF uses this information to update all estimated quantities except the map's state and Cholesky factor (Sect. IV-D – IV-F).

III. System State and Measurement Models

A. Device State

At time-step k, the estimated state is:[5]

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_E^T & \mathbf{x}_{C_{k-1}}^T & \cdots & \mathbf{x}_{C_{k-N}}^T & \mathbf{x}_\tau^T \end{bmatrix}^T \quad (1)$$

where $\mathbf{x}_E$ is the evolving state of the device:

$$\mathbf{x}_E = \begin{bmatrix} {}^{I_k}\mathbf{q}_G^T & \mathbf{b}_{g_k}^T & {}^G\mathbf{v}_{I_k}^T & \mathbf{b}_{a_k}^T & {}^G\mathbf{p}_{I_k}^T \end{bmatrix}^T \quad (2)$$

[4]Note that we use the inverse-depth parameterization w.r.t. the first camera observing a particular point feature.

[5]To simplify the ensuing derivations, we assume the camera and IMU are time synchronized and co-located. In practice, we estimate their extrinsics and time offset following the approaches of [33] and [29], respectively.

${}^{I_k}\mathbf{q}_G$ is the quaternion representation of the global, $\{G\}$, frame's orientation in the IMU's current frame, $\{I_k\}$, $\mathbf{b}_{a_k}$ and $\mathbf{b}_{g_k}$ are the accelerometer and gyroscope biases, respectively, and ${}^G\mathbf{v}_{I_k}$ and ${}^G\mathbf{p}_{I_k}$ are the velocity and position of $\{I_k\}$ in $\{G\}$. In (1), $\mathbf{x}_{C_i} = \begin{bmatrix} {}^{I_i}\mathbf{q}_G^T & {}^G\mathbf{p}_{I_i}^T \end{bmatrix}^T$, $i = k-N, \ldots, k-1$ corresponds to previous IMU poses. Following [3], we maintain a sliding window of N such poses so as to process measurements to non-mapped (or local) features without incorporating them into the state vector.

Finally, our problem formulation requires estimating the 4 degree of freedom (d.o.f.) transformation between the device's global frame, $\{G\}$, and one or more map's frames of reference, $\{M_i\}$, all of which are gravity aligned:

$$\mathbf{x}_\tau = \begin{bmatrix} \mathbf{x}_{\tau_1}^T & \mathbf{x}_{\tau_2}^T & \ldots & \mathbf{x}_{\tau_L}^T \end{bmatrix}^T, \quad \mathbf{x}_{\tau_i} = \begin{bmatrix} {}^{M_i}\phi_G & {}^G\mathbf{p}_{M_i}^T \end{bmatrix}^T \tag{3}$$

where ${}^G\mathbf{p}_{M_i}$ is the position of $\{M_i\}$ in $\{G\}$, and ${}^{M_i}\phi_G$ is the rotation about gravity between the two frames. Note that since the roll and pitch angles are observable for any vision-aided inertial navigation system (VINS) [13], we choose the z-axis to align with gravity for all frames involved.

B. Measurement models

1) IMU measurement model: Following [3], we propagate the state estimate of the device [see (1)] by integrating the IMU's rotational velocity and linear acceleration measurements, $\mathbf{u}_k$,

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{4}$$

where $\mathbf{g}$ is a nonlinear function corresponding to the IMU measurement model and $\mathbf{w}_k$ is zero-mean, Gaussian noise of known covariance, $\mathbf{Q}$ [34].

2) Local-feature measurement model: As the device traverses its environment, it observes and tracks point features that have *not* been mapped. These local features are used to provide measurement constraints between the $N+1$ IMU-camera poses maintained in the state vector. The corresponding non-linear and linearized measurement models (for simplicity, without time indices) are:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \mathbf{p}_f) + \mathbf{n}\,, \qquad \mathbf{r} = \mathbf{H}_R\tilde{\mathbf{x}}_R + \mathbf{H}_f\tilde{\mathbf{p}}_f + \mathbf{n} \tag{5}$$

where $\mathbf{z}$ is the measurement, $\mathbf{h}$ is the perspective-projection camera measurement model, $\mathbf{r}$ is the linearized measurement residual, $\mathbf{x}_R$ ($\tilde{\mathbf{x}}_R$) is the device state (error-state), $\mathbf{p}_f$ ($\tilde{\mathbf{p}}_f$) is the local feature state (error-state), $\mathbf{H}_R$ and $\mathbf{H}_f$ are the measurement Jacobians corresponding to the device and feature states, respectively, and $\mathbf{n}$ is zero-mean, Gaussian noise with known covariance, $\mathbf{R}$.

As in [3], we marginalize the local feature by projecting $\mathbf{r}$ on the left null space of $\mathbf{H}_f$, $\mathbf{U}$:

$$\mathbf{r}^o = \mathbf{H}_R^o\tilde{\mathbf{x}}_R + \mathbf{n}^o \tag{6}$$

where:

$$\mathbf{r}^o \triangleq \mathbf{U}^T\mathbf{r}\,, \qquad \mathbf{H}_R^o \triangleq \mathbf{U}^T\mathbf{H}_R\,, \qquad \mathbf{n}^o \triangleq \mathbf{U}^T\mathbf{n}$$

The new measurement residual, $\mathbf{r}^o$, and Jacobian, $\mathbf{H}_R^o$, are then (Sect. IV-C.2) used for updating the device state estimate.

3) Mapped-feature measurement model: When a previously-mapped point-feature, f, (expressed w.r.t. the IMU-camera frame, $\{I_\lambda^{M_i}\}$, that first observed it during mapping) is detected by the device, the following geometric relation holds (see Fig. 1):

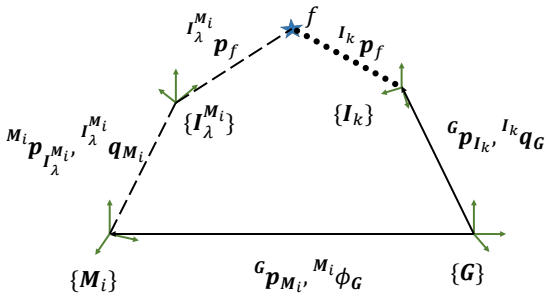

Fig. 1. Mapped-feature observation: The dotted line is a bearing measurement, solid lines correspond to quantities estimated online. Dashed lines correspond to variables determined during mapping offline.

$${}^{I_k}\mathbf{p}_f = {}^{I_k}_G\mathbf{C}\left({}^G\mathbf{p}_{M_i} - {}^G\mathbf{p}_{I_k} + {}^G_{M_i}\mathbf{C}\left[{}^{M_i}\mathbf{p}_{I_\lambda^{M_i}} + {}^{M_i}_{I_\lambda^{M_i}}\mathbf{C}\;{}^{I_\lambda^{M_i}}\mathbf{p}_f\right]\right) \tag{7}$$

where all rotation matrices, $\mathbf{C}$, are parameterized by their corresponding quaternions, except ${}^G_{M_i}\mathbf{C}$, which corresponds to a rotation about gravity by an angle ${}^{M_i}\phi_G$. Applying the camera perspective-projection transformation, $\boldsymbol{\pi}$, leads to the following measurement model:

$$\mathbf{z}_{map} = \boldsymbol{\pi}\left({}^{I_k}\mathbf{p}_f\right) + \mathbf{n} = \mathbf{h}_{map}(\mathbf{x}_k, {}^{I_\lambda^{M_i}}\mathbf{p}_f) + \mathbf{n} \tag{8}$$

where $\mathbf{n}$ is zero-mean, Gaussian noise with covariance $\mathbf{R}$.

To simplify the notation, from here on, we denote the state $\mathbf{x}_k$ in (1) by $\mathbf{x}_R$, while we use $\mathbf{x}_M$ to represent the vector comprising all mapped features and corresponding IMU-camera poses. Linearizing (8) yields:

$$\mathbf{r} = \mathbf{H}_R\tilde{\mathbf{x}}_R + \mathbf{H}_M\tilde{\mathbf{x}}_M + \mathbf{n} \tag{9}$$

where $\mathbf{H}_R$ and $\mathbf{H}_M$ are the device and map Jacobians, respectively. Note that both $\mathbf{H}_R$ and $\mathbf{H}_M$ are sparse, as the measurement equation only involves the pose of the current and mapped IMU-camera pairs, the 4 d.o.f. map-to-global transformation, and the feature's position.

IV. Algorithm Description

In what follows, we first review the SKF, and then introduce the C-SKF and sC-SKF. To simplify notation, we denote updated and propagated quantities with $\oplus$ and $\ominus$, respectively, rather than using time subscripts. [i.e., $\mathbf{P}_{k+1|k+1} = \mathbf{f}(\mathbf{P}_{k+1|k})$ is expressed as $\mathbf{P}^\oplus = \mathbf{f}(\mathbf{P})$, and $\mathbf{P}_{k+1|k} = \mathbf{g}(\mathbf{P}_{k|k})$ is $\mathbf{P}^\ominus = \mathbf{g}(\mathbf{P})$]

A. Background: Schmidt-Kalman filter

Consider the current propagated system covariance, $\mathbf{P}$, and Jacobian, $\mathbf{H}$, to be partitioned as:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{RR} & \mathbf{P}_{RM} \\ \mathbf{P}_{MR} & \mathbf{P}_{MM} \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} \mathbf{H}_R & \mathbf{H}_M \end{bmatrix} \tag{10}$$

The state and covariance update equations for the extended Kalman filter (EKF) are:

$$\hat{\mathbf{x}}^\oplus = \hat{\mathbf{x}} + \mathbf{K}\mathbf{r} \tag{11}$$

$$\mathbf{P}^\oplus = (\mathbf{I} - \mathbf{K}\mathbf{H})\,\mathbf{P}\,(\mathbf{I} - \mathbf{H}^T\mathbf{K}^T) + \mathbf{K}\mathbf{R}\mathbf{K}^T \tag{12}$$

where

$$\mathbf{r} = \mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}), \quad \mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}$$
$$\mathbf{K} = \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1} = \begin{bmatrix} \mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{P}_{RM}\mathbf{H}_M^T \\ \mathbf{P}_{MR}\mathbf{H}_R^T + \mathbf{P}_{MM}\mathbf{H}_M^T \end{bmatrix} \mathbf{S}^{-1} = \begin{bmatrix} \bar{\mathbf{K}}_R \\ \bar{\mathbf{K}}_M \end{bmatrix} \mathbf{S}^{-1} \tag{13}$$

As shown in [9], the SKF updates only part of state by zeroing the Kalman gain associated with the state elements whose estimates are to remain the same (in our case the map, $\mathbf{x}_M$):

$$\mathbf{K}_{SKF} = \begin{bmatrix} (\bar{\mathbf{K}}_R\mathbf{S}^{-1})^T & \mathbf{0} \end{bmatrix}^T \tag{14}$$

Substituting (14) in (11) yields the following state update:

$$\hat{\mathbf{x}}_R^{\oplus} = \hat{\mathbf{x}}_R + \bar{\mathbf{K}}_R\mathbf{S}^{-1}\mathbf{r} \qquad \hat{\mathbf{x}}_M^{\oplus} = \hat{\mathbf{x}}_M \tag{15}$$

Additionally, by employing (14) in (12), we arrive at the SKF's covariance update:

$$\mathbf{P}_{SKF}^{\oplus} = \mathbf{P} - \begin{bmatrix} \bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_R^T & \bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_M^T \\ \bar{\mathbf{K}}_M\mathbf{S}^{-1}\bar{\mathbf{K}}_R^T & \mathbf{0} \end{bmatrix} \tag{16}$$

Note that if we express the updated covariance of the EKF as a function of the SKF updated covariance, it is straightforward to show the SKF is consistent:

$$\mathbf{P}_{EKF}^{\oplus} = \mathbf{P}_{SKF}^{\oplus} - \begin{bmatrix} \mathbf{0} & \bar{\mathbf{K}}_M^T \end{bmatrix}^T \mathbf{S}^{-1} \begin{bmatrix} \mathbf{0} & \bar{\mathbf{K}}_M^T \end{bmatrix} \preceq \mathbf{P}_{SKF}^{\oplus} \tag{17}$$

since $\begin{bmatrix} \mathbf{0} & \bar{\mathbf{K}}_M^T \end{bmatrix}^T \mathbf{S}^{-1} \begin{bmatrix} \mathbf{0} & \bar{\mathbf{K}}_M^T \end{bmatrix}$ is positive semi-definite.

Furthermore, as evident from (16), the cost of an SKF update is linear in the size of the map, as only the device's state, covariance, and cross-correlation terms need to be updated, while $\mathbf{H}$ in (10) is sparse. On the other hand, the SKF requires storing the covariance of the map, $\mathbf{P}_{MM}$, which is dense. To better appreciate the challenge this poses on mobile devices, we note that the covariance of a map generated from 3.5 min of visual and inertial data requires over 2 GB of storage space. Instead, the sparse Cholesky factor of the corresponding Hessian matrix requires only 107.3 MB, and after sparsification (Sect. V) can be further reduced to 50.4 MB. This motivates us to introduce the Cholesky-SKF (C-SKF) that provides the same consistency guarantees as the SKF [see (17)] while significantly reducing the memory requirements.

B. C-SKF device-map initialization

We start by describing the process for estimating the 4 d.o.f. transformation between the map's frame and the device's global reference frame, as well as its covariance and correlations with all estimated quantities.

Specifically, consider the first time the mobile device observes two or more previously-mapped features. An initial estimate, $\hat{\mathbf{x}}_\tau$, for the 4 d.o.f. transformation is obtained by employing the 2+1 pt RANSAC [35]. Furthermore, we partition the current state as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{R'}^T & \mathbf{x}_\tau^T & \mathbf{x}_M^T \end{bmatrix}^T \tag{18}$$

where $\mathbf{x}_{R'}$ comprises of the remaining elements of the device's state vector [see (1)], and the corresponding covariance as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{R'R'} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\tau\tau} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix}, \mathbf{P}_{\tau\tau} = \lim_{\mu\to\infty} (\mu\mathbf{I}_4) \tag{19}$$

where $\mathbf{P}_{\tau\tau}$ is the covariance of the unknown 4 d.o.f. transformation, $\mathbf{G}\mathbf{G}^T$ is the Hessian of the map, and $\mathbf{G}$ is its Cholesky factor. After linearizing the measurement model in (8) and denoting the corresponding Jacobian as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{R'} & \mathbf{H}_\tau & \mathbf{H}_M \end{bmatrix} \tag{20}$$

we employ (13)-(15) to update the estimates of $\mathbf{x}$:

$$\hat{\mathbf{x}}_{R'}^{\oplus} = \hat{\mathbf{x}}_{R'} + \mathbf{P}_{R'R'}\mathbf{H}_{R'}^T\mathbf{S}^{-1}\mathbf{r} \tag{21}$$
$$\hat{\mathbf{x}}_\tau^{\oplus} = \hat{\mathbf{x}}_\tau + (\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{H}_\tau)^{-1}\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{r} \tag{22}$$
$$\hat{\mathbf{x}}_M^{\oplus} = \hat{\mathbf{x}}_M \tag{23}$$

Additionally, employing (16), it can be shown that the updated SKF covariance is

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{R'R'}^{\oplus} & \mathbf{P}_{R'\tau}^{\oplus} & \mathbf{P}_{R'M}^{\oplus} \\ \mathbf{P}_{R'\tau}^{\oplus T} & \mathbf{P}_{\tau\tau}^{\oplus} & \mathbf{P}_{\tau M}^{\oplus} \\ \mathbf{P}_{R'M}^{\oplus T} & \mathbf{P}_{\tau M}^{\oplus T} & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{RR}^{\oplus} & \mathbf{P}_{RM}^{\oplus} \\ \mathbf{P}_{MR}^{\oplus} & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix} \tag{24}$$

where

$$\begin{aligned} \mathbf{P}_{R'R'}^{\oplus} &= \mathbf{P}_{R'R'} - \mathbf{P}_{R'R'}\mathbf{H}_{R'}^T\mathbf{S}^{-1}\mathbf{H}_{R'}\mathbf{P}_{R'R'} \\ \mathbf{P}_{R'\tau}^{\oplus} &= -\mathbf{P}_{R'R'}\mathbf{H}_{R'}^T\mathbf{A}^{-1}\mathbf{H}_\tau(\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{H}_\tau)^{-1} \\ \mathbf{P}_{\tau\tau}^{\oplus} &= (\mathbf{H}_\tau^T\mathbf{A}\mathbf{H}_\tau)^{-1} \\ \mathbf{P}_{RM}^{\oplus} &= \begin{bmatrix} \mathbf{P}_{R'M}^{\oplus} \\ \mathbf{P}_{\tau M}^{\oplus} \end{bmatrix} = \begin{bmatrix} -\mathbf{P}_{R'R'}\mathbf{H}_{R'}^T\mathbf{S}^{-1}\mathbf{J} \\ (\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{H}_\tau)^{-1}\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{J} \end{bmatrix} \mathbf{G}^{-1} \\ &= \mathbf{\Gamma}\mathbf{G}^{-1} \end{aligned} \tag{25}$$

and

$$\begin{aligned} \mathbf{S}^{-1} &= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_\tau(\mathbf{H}_\tau^T\mathbf{A}^{-1}\mathbf{H}_\tau)^{-1}\mathbf{H}_\tau^T\mathbf{A}^{-1} \\ \mathbf{A} &= \mathbf{H}_{R'}\mathbf{P}_{R'R'}\mathbf{H}_{R'} + \mathbf{J}\mathbf{J}^T + \mathbf{R} \end{aligned}$$

Finally, $\mathbf{J}$ is defined as:

$$\mathbf{G}\mathbf{J}^T = \mathbf{H}_M^T \tag{26}$$

A key element of our approach is that, from this point on, instead of updating the cross-correlation term, $\mathbf{P}_{RM}$, we will represent it in a factorized form [see (25)] and apply updates on its factor, $\mathbf{\Gamma}$, as is shown in Sect. IV-C and Sect. IV-D. Following this convention, we have:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{RR} & \mathbf{\Gamma}\mathbf{G}^{-1} \\ (\mathbf{\Gamma}\mathbf{G}^{-1})^T & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix} \tag{27}$$

Note that we do *not* compute the inverse of the Cholesky factor, $\mathbf{G}$. Instead, all update equations involving $\mathbf{G}$ will be of the same form as (26), where a back-solve involving the sparse $\mathbf{G}$ is required for efficiently computing $\mathbf{J}$.

C. C-SKF propagation and local-feature update

1) IMU-based Propagation: By employing the IMU measurement model in (4), the device's state is propagated while, as is the case for the SKF, the map's state remains the same:

$$\hat{\mathbf{x}}_R^{\ominus} = \mathbf{f}(\hat{\mathbf{x}}_R, \mathbf{u}), \qquad \hat{\mathbf{x}}_M^{\ominus} = \mathbf{x}_M \tag{28}$$

On the other hand, and in order to propagate the covariance of the C-SKF [see (27)], we employ the EKF covariance

propagation equation:

$$\mathbf{P}^{\ominus} = \mathbf{\Phi}_C \mathbf{P} \mathbf{\Phi}_C^T + \mathbf{Q}_C = \begin{bmatrix} \mathbf{\Phi}\mathbf{P}_{RR}\mathbf{\Phi}^T + \mathbf{Q} & \mathbf{\Phi}\mathbf{\Gamma}\mathbf{G}^{-1} \\ (\mathbf{\Phi}\mathbf{\Gamma}\mathbf{G}^{-1})^T & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix} \quad (29)$$

with

$$\mathbf{\Phi}_C = \begin{bmatrix} \mathbf{\Phi} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \qquad \mathbf{Q}_C = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (30)$$

where $\mathbf{\Phi}$ and $\mathbf{Q}$ are the IMU Jacobian of the state and corresponding noise covariance, respectively.

2) C-SKF Local-Feature Measurement Update: When a local-feature-track measurement becomes available [see (6)], we arrive at the following Jacobian:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_R^o & \mathbf{0} \end{bmatrix} \quad (31)$$

As in (14), we zero out the Kalman gain associated with the mapped states, leading to an update of the device state, while the map remains the same:

$$\hat{\mathbf{x}}_R^{\oplus} = \hat{\mathbf{x}}_R + \mathbf{P}_{RR}\mathbf{H}_R^{oT}\mathbf{S}^{-1}\mathbf{r}\,, \qquad \hat{\mathbf{x}}_M^{\oplus} = \hat{\mathbf{x}}_M \quad (32)$$

For the covariance update, we first denote $\bar{\mathbf{K}} = \mathbf{P}\mathbf{H}^T$:

$$\begin{bmatrix} \bar{\mathbf{K}}_R \\ \bar{\mathbf{K}}_M \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{RR} & \mathbf{\Gamma}\mathbf{G}^{-1} \\ (\mathbf{\Gamma}\mathbf{G}^{-1})^T & (\mathbf{G}\mathbf{G}^T)^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{H}_R^{oT} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{RR}\mathbf{H}_R^{oT} \\ \mathbf{G}^{-T}\mathbf{\Gamma}^T\mathbf{H}_R^{oT} \end{bmatrix} \quad (33)$$

Next, we employ (33) and (27) in (16) without evaluating $\bar{\mathbf{K}}_M$, yielding:

$$\mathbf{P}_{RR}^{\oplus} = \mathbf{P}_{RR} - \mathbf{P}_{RR}\mathbf{H}_R^{oT}\mathbf{S}^{-1}\mathbf{H}_R^o\mathbf{P}_{RR}\,, \qquad \mathbf{P}_{MM}^{\oplus} = \mathbf{P}_{MM} \quad (34)$$

and

$$\begin{aligned} \mathbf{P}_{RM}^{\oplus} &= \mathbf{\Gamma}\mathbf{G}^{-1} - \bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_M^T \\ &= \left(\mathbf{I} - \mathbf{P}_{RR}\mathbf{H}_R^{oT}\mathbf{S}^{-1}\mathbf{H}_R^o\right)\mathbf{\Gamma}\mathbf{G}^{-1} = \mathbf{\Gamma}^{\oplus}\mathbf{G}^{-1} \end{aligned} \quad (35)$$

As evident from (29) and (35), both propagation and local-feature updates have complexity *linear* in the size of the map, as we only need to apply a standard EKF update on the device's covariance, and update the cross-correlation factor, $\mathbf{\Gamma}$.

D. C-SKF map-based updates

When the device observes previously mapped features, we employ the methodology of Sect. IV-A [(13)–(16)] using the measurement model of (8)–(9), and operate on the system covariance with factorized cross-correlation, as defined in (27). Specifically, we denote $\bar{\mathbf{K}} = \mathbf{P}\mathbf{H}^T$ as:

$$\begin{bmatrix} \bar{\mathbf{K}}_R \\ \bar{\mathbf{K}}_M \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{\Gamma}\mathbf{G}^{-1}\mathbf{H}_M^T \\ \mathbf{G}^{-T}\mathbf{\Gamma}^T\mathbf{H}_R^T + (\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{H}_M^T \end{bmatrix} \quad (36)$$

Note that we do *not* explicitly compute $\bar{\mathbf{K}}_M$, instead we first compute $\mathbf{J}$ as:

$$\mathbf{G}\mathbf{J}^T = \mathbf{H}_M^T \quad (37)$$

Because $\mathbf{G}$ is triangular, we can compute $\mathbf{J}$ with a back-solve operation. Substituting $\mathbf{J}$ into (36) yields:

$$\begin{bmatrix} \bar{\mathbf{K}}_R \\ \bar{\mathbf{K}}_M \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{\Gamma}\mathbf{J}^T \\ \mathbf{G}^{-T}\mathbf{\Gamma}^T\mathbf{H}_R^T + \mathbf{G}^{-T}\mathbf{J}^T \end{bmatrix} \quad (38)$$

Next, we compute the residual covariance:

$$\mathbf{S} = \mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{H}_R\mathbf{\Gamma}\mathbf{J}^T + \mathbf{J}\mathbf{\Gamma}^T\mathbf{H}_R^T + \mathbf{J}\mathbf{J}^T + \mathbf{R} \quad (39)$$

and state update [see (15)]:

$$\hat{\mathbf{x}}_R^{\oplus} = \hat{\mathbf{x}}_R + \bar{\mathbf{K}}_R\mathbf{S}^{-1}\mathbf{r}\,, \qquad \hat{\mathbf{x}}_M^{\oplus} = \hat{\mathbf{x}}_M \quad (40)$$

Finally, we update the covariance using (16) with (38), and following the same process as in (35), we produce a factorized form of the updated cross-correlation:

$$\begin{aligned} \mathbf{P}_{RR}^{\oplus} &= \mathbf{P}_{RR} - \bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_R^T \\ \mathbf{P}_{RM}^{\oplus} &= \mathbf{\Gamma}\mathbf{G}^{-1} - \bar{\mathbf{K}}_R\mathbf{S}^{-1}(\mathbf{H}_R\mathbf{\Gamma}\mathbf{G}^{-1} + \mathbf{J}\mathbf{G}^{-1}) \\ &= [\mathbf{\Gamma} - \bar{\mathbf{K}}_R\mathbf{S}^{-1}(\mathbf{H}_R\mathbf{\Gamma} + \mathbf{J})]\mathbf{G}^{-1} = \mathbf{\Gamma}^{\oplus}\mathbf{G}^{-1} \end{aligned} \quad (41)$$

At this point, we should note that unlike propagation and local-feature updates, the processing requirements of mapped-feature updates are not strictly linear in the size of the map. The main bottleneck is (37), as computing $\mathbf{J}$ has complexity between linear and quadratic in the size of the map (depending on the structure of $\mathbf{G}$). As the map grows, the time to compute $\mathbf{J}$ becomes unacceptable for real-time operation on a mobile device. This limitation motivates us to introduce two consistent relaxations of the C-SKF: (i) The sub-map relaxation of Sect. IV-E and Sect. IV-F. That is, we decrease the size of $\mathbf{G}$ in (37) by partitioning the map into independent sub-maps. (ii) The consistent sparsification of a given map, as described in Sect. V, which removes off-diagonal elements in $\mathbf{G}$ caused by loop-closure measurements.

E. Sub-map relaxation

Before discussing the sC-SKF, we first describe the sub-mapping relaxation process. As shown in Fig. 2 and described below, we divide the trajectory and associated features into two separate sets:[6]

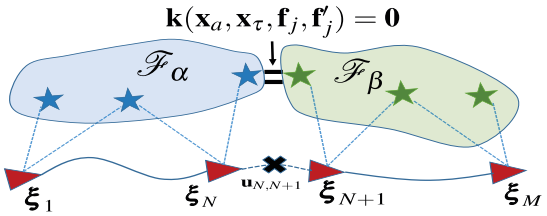

Fig. 2. A partitioning of a single map into two sub-maps, with a geometric constraint, $\mathbf{k}(\mathbf{x}_a, \mathbf{x}_\tau, \mathbf{f}_j, \mathbf{f}'_j) = \mathbf{0}$, imposed to common features.

First, the IMU-camera poses, $\boldsymbol{\xi}_i$, $i = 1 \ldots M$, are divided into $[\boldsymbol{\xi}_1, \boldsymbol{\xi}_N]$ and $[\boldsymbol{\xi}_{N+1}, \boldsymbol{\xi}_M]$.[7] With this division defined, we partition features into two sets: (i) $\mathscr{F}_\alpha$: those observed by poses $\boldsymbol{\xi}_1$ to $\boldsymbol{\xi}_N$, and (ii) $\mathscr{F}'_\beta$: those observed by poses $\boldsymbol{\xi}_{N+1}$ to $\boldsymbol{\xi}_M$. Features in $\mathscr{F}_c = \mathscr{F}_\alpha \cap \mathscr{F}'_\beta$ are common features observed in both sub-maps. The cooperative mapping (CM) algorithm [17] creates a "duplicate" feature set of $\mathscr{F}_c$, $\mathscr{F}'_c$, and assigns these duplicate features to the second sub-map [i.e., $\mathscr{F}_\beta = (\mathscr{F}'_\beta - \mathscr{F}_c) \cup \mathscr{F}'_c$] such that the two sub-map's individual cost functions are independent [see (43)]. Finally, CM introduces a non-linear geometric constraint between the features in $\mathscr{F}_c$, $\mathbf{f}_j$, and their duplicates, $\mathbf{f}'_j$:

$$\mathbf{k}(\mathbf{x}_a, \mathbf{x}_\tau, \mathbf{f}_j, \mathbf{f}'_j) = \mathbf{0}, \;\; \mathbf{f}_j \in \mathscr{F}_c, \mathbf{f}'_j \in \mathscr{F}'_c \quad (42)$$

[6]Without loss of generality, we present the two sub-map case

[7]The current partitioning distributes these sets evenly in time.

where $\mathbf{x}_a$ is the state of all IMU-camera poses and $\mathbf{x}_\tau$ is the 4 d.o.f. transformation between the two sub-maps. For each common feature in $\mathscr{F}_\alpha$, this constraint enforces its corresponding "duplicate" in $\mathscr{F}_\beta$ to have the same physical position. Finally, all camera and IMU measurements, $\mathbf{z}_{i,j}$ and $\mathbf{u}_{i,i+1}$, are assigned to their corresponding sub-map (with the exception of $\mathbf{u}_{N,N+1}$, which is discarded).

With such a partitioning, by ignoring the common-feature constraints, we form two independent cost functions corresponding to each sub-map, $\mathscr{C}_1$ and $\mathscr{C}_2$:

$$\mathscr{C}_1 = \sum_{\substack{i=1\\ \mathbf{f}_j\in\mathscr{F}_\alpha}}^{N} ||\mathbf{z}_{i,j}-\mathbf{h}(\boldsymbol{\xi}_i,\mathbf{f}_j)||^2_{\mathbf{R}} + \sum_{i=1}^{N-1} ||\boldsymbol{\xi}_{i+1}-\mathbf{g}(\boldsymbol{\xi}_i,\mathbf{u}_{i,i+1})||^2_{\mathbf{Q}}$$
$$\mathscr{C}_2 = \sum_{\substack{i=N+1\\ \mathbf{f}_j\in\mathscr{F}_\beta}}^{M} ||\mathbf{z}_{i,j}-\mathbf{h}(\boldsymbol{\xi}_i,\mathbf{f}_j)||^2_{\mathbf{R}} + \sum_{i=N+1}^{M-1} ||\boldsymbol{\xi}_{i+1}-\mathbf{g}(\boldsymbol{\xi}_i,\mathbf{u}_{i,i+1})||^2_{\mathbf{Q}} \tag{43}$$

where $\mathbf{g}$ and $\mathbf{h}$ are defined in (4) and (5), respectively. Summing these two cost functions and imposing the common-feature constraints yields the following optimization problem:

$$\mathbf{x}_a^*,\mathbf{x}_\tau^*,\mathscr{F}_\alpha^*,\mathscr{F}_\beta^* = \operatorname{argmin}\, \mathscr{C}_1+\mathscr{C}_2 \tag{44}$$
$$\text{s. t.}\quad \mathbf{k}(\mathbf{x}_a,\mathbf{x}_\tau,\mathbf{f}_j,\mathbf{f}'_j)=\mathbf{0},\ \mathbf{f}_j\in\mathscr{F}_c,\mathbf{f}'_j\in\mathscr{F}'_c$$

which we solve offline by employing CM. At this point, we should note that, as shown in [17], the solution of (44) is the same as that of the original BLS of the single map problem (after discarding $\mathbf{u}_{N,N+1}$). Thus, the estimated feature positions will be as accurate as if no map partitioning had occured. Furthermore, at each iteration of Gauss-Newton minimization of (44), the Cholesky factors $\mathbf{G}_1$ and $\mathbf{G}_2$ of the Hessians [corresponding to $\mathscr{C}_1$ and $\mathscr{C}_2$ in (43)] of the two submaps are computed, and hence are available to be used later on (see Sect. IV-F) for reformulating consistent map-based updates within the C-SKF framework.

F. Cholesky-Kalman-Schmidt with sub-maps (sC-SKF)

In our problem, we take advantage of the high-accuracy state estimates computed from the solution of (44), but relax the information attributed to each sub-map by storing the Cholesky factors, $\mathbf{G}_i$, resulting from each corresponding cost function in (43) linearized at the CM solution of (44).[8]

To process mapped measurements with sub-maps, we represent the system covariance as:

$$\mathbf{P}=\begin{bmatrix}\mathbf{P}_{RR} & \boldsymbol{\Gamma}_1\mathbf{G}_1^{-1} & \boldsymbol{\Gamma}_2\mathbf{G}_2^{-1}\\ (\boldsymbol{\Gamma}_1\mathbf{G}_1^{-1})^T & (\mathbf{G}_1\mathbf{G}_1^T)^{-1} & \mathbf{0}\\ (\boldsymbol{\Gamma}_2\mathbf{G}_2^{-1})^T & \mathbf{0} & (\mathbf{G}_2\mathbf{G}_2^T)^{-1}\end{bmatrix} \tag{45}$$

where $\boldsymbol{\Gamma}_i$ and $\mathbf{G}_i$ are the cross-correlation factor from (27) and the Cholesky factor of the i-th sub-map, respectively.

When a mapped feature in, e.g., the first sub-map is observed, the measurement Jacobian is:

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}_R & \mathbf{H}_1 & \mathbf{0}\end{bmatrix} \tag{46}$$

where $\mathbf{H}_1$ corresponds to the states in the first sub-map. Following (37) and (38), we compute $\mathbf{J}_1$ and express $\bar{\mathbf{K}}$ as:

$$\mathbf{G}_1\mathbf{J}_1^T=\mathbf{H}_1^T,\qquad \begin{bmatrix}\bar{\mathbf{K}}_R\\ \bar{\mathbf{K}}_1\\ \bar{\mathbf{K}}_2\end{bmatrix}=\begin{bmatrix}\mathbf{P}_{RR}\mathbf{H}_R^T+\boldsymbol{\Gamma}_1\mathbf{J}_1^T\\ \mathbf{G}_1^{-T}(\boldsymbol{\Gamma}_1^T\mathbf{H}_R^T+\mathbf{J}_1^T)\\ \mathbf{G}_2^{-T}\boldsymbol{\Gamma}_2^T\mathbf{H}_R^T\end{bmatrix} \tag{47}$$

The residual covariance and state update are computed as:

$$\mathbf{S}=\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T+\mathbf{H}_R\boldsymbol{\Gamma}_1\mathbf{J}_1^T+\mathbf{J}_1\boldsymbol{\Gamma}_1^T\mathbf{H}_R^T+\mathbf{J}_1\mathbf{J}_1^T+\mathbf{R}$$
$$\hat{\mathbf{x}}_R^{\oplus}=\hat{\mathbf{x}}_R+\bar{\mathbf{K}}_R\mathbf{S}^{-1}\mathbf{r},\qquad \hat{\mathbf{x}}_1^{\oplus}=\hat{\mathbf{x}}_1,\qquad \hat{\mathbf{x}}_2^{\oplus}=\hat{\mathbf{x}}_2 \tag{48}$$

Finally, we update the covariance and cross-correlation using the same factorization as the single-map case [see (41)]:

$$\mathbf{P}_{RR}^{\oplus}=\mathbf{P}_{RR}-\bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_R^T,\qquad \mathbf{P}_{11}^{\oplus}=\mathbf{P}_{11},\qquad \mathbf{P}_{22}^{\oplus}=\mathbf{P}_{22}$$
$$\begin{aligned}\mathbf{P}_{R1}^{\oplus}&=\boldsymbol{\Gamma}_1\mathbf{G}_1^{-1}-\bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_1^T\\ &=[\boldsymbol{\Gamma}_1-\bar{\mathbf{K}}_R\mathbf{S}^{-1}(\mathbf{H}_R\boldsymbol{\Gamma}_1+\mathbf{J}_1)]\mathbf{G}_1^{-1}=\boldsymbol{\Gamma}_1^{\oplus}\mathbf{G}_1^{-1}\end{aligned} \tag{49}$$
$$\begin{aligned}\mathbf{P}_{R2}^{\oplus}&=\boldsymbol{\Gamma}_2\mathbf{G}_2^{-1}-\bar{\mathbf{K}}_R\mathbf{S}^{-1}\bar{\mathbf{K}}_2^T\\ &=[\mathbf{I}-\bar{\mathbf{K}}_R\mathbf{S}^{-1}\mathbf{H}_R]\boldsymbol{\Gamma}_2\mathbf{G}_2^{-1}=\boldsymbol{\Gamma}_2^{\oplus}\mathbf{G}_2^{-1}\end{aligned} \tag{50}$$

Note that partitioning the map into two independent sub-maps significantly lowers the cost of the C-SKF map-based updates. Specifically, by reducing the size of the Cholesky factor involved in the mapped-feature update, we lower the processing requirements of the C-SKF's bottleneck (the back-substitution in (37) for computing $\mathbf{J}$) which has complexity between linear and quadratic in the map's size, depending on the structure of the Cholesky factor. It should be noted, however, that increasing the number of sub-maps decreases the information associated with the map, which typically leads to less accurate device pose estimates.

V. Consistent Map Sparsification

As previously mentioned, the cost of the back-solve required by the C-SKF [see (37)] depends not only on the size of the (sub-)map, but also on the structure of the corresponding Cholesky factor. In particular, if a map does not contain loop-closure (LC) measurements, the resulting Hessian forms a banded structure, which in turn yields a triangular-banded Cholesky factor. In the presence of many loop-closures, however, the off-diagonal of the Cholesky factor becomes dense, causing both the memory requirements and computational complexity of the back-solve step to increase from *linear*, in the size of the map, to *quadratic*. Our approach to address this issue is to consider the contribution of only a subset of select LC measurements in the Hessian matrix used for computing the Cholesky factor.[9] Specifically, the Hessian of the map can be written as:

$$\mathscr{H}=\mathscr{H}'-\mathbf{H}_m^T\mathbf{H}_m \tag{51}$$

Where $\mathscr{H}'$ is the original map's Hessian (using all measurements), and $\mathbf{H}_m$ is the whitened Jacobian corresponding to all neglected LC measurements. It is easy to see that this relaxation is consistent (i.e., $\mathscr{H}\preceq\mathscr{H}'$), because $\mathbf{H}_m^T\mathbf{H}_m$ is positive semi-definite by construction.

[8] Such a relaxation causes the sub-maps to become independent, but, as show in [32], maintains consistency, since it corresponds to dropping the information associated with the geometric constraint $\mathbf{k}$ in (44).

[9] Note that the method described here only affects the calculation of the (sub-)map's Cholesky factor. For the mapped poses' and features' state estimates, we use the result of solving the original optimization problem in (44).

When selecting LC measurements to drop, we attempt to satisfy the following objectives: (i) The Cholesky factor is sufficiently sparse; (ii) The information loss due to this relaxation is minimal; (iii) The measurements to be dropped can be efficiently determined.

To do so we introduce a heuristic, linear (in the number of mapped poses) complexity algorithm. In particular, we consider each pose in the mapped-trajectory in temporal order:

1) If LC measurements to this pose exist, we keep (drop) them if the time since the previous LC is above (below) a threshold, t_ℓ (e.g., $t_\ell = 2$ min).
2) Else (no LCs observed) if at least t_ℓ has passed since the last kept LC, we check if there has been a pose with LC measurements in the near past (within a threshold $t_s = 30$ sec), and retroactively enable them.

Once each mapped-pose has been visited, we reform the map's Hessian and its Cholesky factor using only the kept LC measurements (in addition to the consecutive-feature tracks and IMU measurements).

The intuition behind this approach is that the value of a LC (in terms of information gain) grows the longer the user has been exploring without closing a loop. Thus, we set a threshold for the time between consecutive LCs, and drop any that are encountered until this time threshold is reached. On the other hand, step (2) ensures we do not miss an important LC before an extended exploration phase. Such a strategy satisfies the objectives given earlier: (i) We can control the sparsity of the Cholesky factor by tuning the thresholds t_ℓ and t_s (ii) LC measurements that occur after extended periods of exploration, which cause large corrections are retained, while subsequent, close in space and time, measurements, whose impact is small, are neglected. (iii) The selection process is very fast, only needing to visit each pose in the mapped trajectory once.

VI. Experimental Results

All experimental results are obtained on a Google Tango tablet [36], which is equipped with a fisheye, global-shutter, gray-scale camera, a MEMS quality IMU, a quad-core, 2.3 GHz ARM Cortex-A15 CPU, and 4 GB of RAM.

The accuracy of the sC-SKF and C-SKF with and without map sparisification is reported in Table I, and compared to the inconsistent map-based localization method of treating the map as perfect, while inflating the measurement noise (e.g., [3], [4]). All four datasets are indoors and of varying sizes. The smallest one is limited to a single room with VICON ground truth. The other three correspond to large buildings, include challenging motions (moving along the optical axis through open spaces) under unfavorable lighting conditions (several underexposed scenes), and are evaluated against BLS ground truth. It should be noted that for significant portions (40%) of the 3-floor long dataset, the user navigates throughout unmapped areas, which leads to larger errors during these periods. In order to compare the consistency of our method against that of [3] and [4], Fig. 3 shows the error and 3σ bounds for the sC-SKF estimator employing 2 sub-maps and for the inflated-measurement-noise approach ($\sigma = 7.5$ pixels), in the room dataset. Note that in the latter case the errors in y exceed the 3σ bounds.

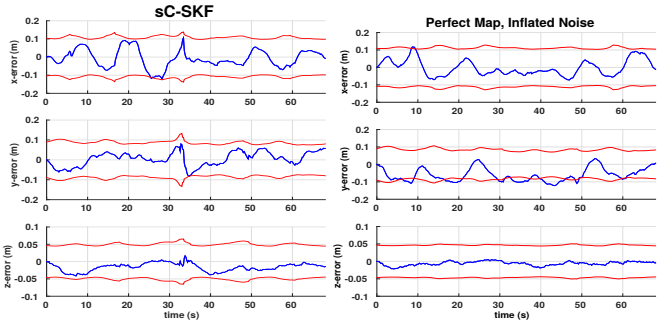

Fig. 3. Position error and 3σ bounds for the sC-SKF with 2 sub-maps (left) and the perfect map approximation (right).

Dataset	room	single-floor	3-floor short	3-floor long
Num. Mapped Features	2.2K	8.7K	15.8K	15.8K
Num. Submaps (for sC-SKF)	2	4	6	6
Trajectory Length	71 m	293 m	198 m	744 m
C-SKF	6.2 cm	13.1 cm	8.1 cm	49.5 cm
Sparse-map C-SKF	6.4 cm	13.3 cm	8.0 cm	50.4 cm
sC-SKF	6.6 cm	13.4 cm	8.5 cm	51.2 cm
Sparse-map sC-SKF	6.9 cm	15.8 cm	8.7 cm	51.6 cm
Perfect Map Approx.	8.3 cm	18.3 cm	10.5 cm	57.6 cm

TABLE I

RMSE COMPARISON OF THE PROPOSED METHODS AGAINST THE PERFECT MAP APPROXIMATION.

Num. Feat.	2,095	2,397	3,433	8,770
P Size	158 MB	207 MB	425 MB	2.8 GB
G Size	38.9 MB	22.0 MB	161.4 MB	107.3 MB
$\mathbf{G}_{\text{sparse}}$ Size	26.5 MB	15.6 MB	32.1 MB	50.4 MB
Update Time, (**G**)	162.4 ms	140.4 ms	427.5 ms	712.8 ms
Update Time, ($\mathbf{G}_{\text{sparse}}$)	150.7 ms	132.4 ms	184.5 ms	501.9 ms

TABLE II

MEMORY REQUIREMENTS AND UPDATE TIMINGS OF VARIOUS MAPS

Furthermore, the results of Table I demonstrate that the C-SKF outperforms the inconsistent perfect map approximation, even after employing the sub-map and sparsification relaxations.

As mentioned earlier, the storage requirements of the map's uncertainty (or information) is the main limitation of the SKF addressed by the C-SKF, and further improved by sparsification. The sizes of the covariance, **P**, (required by the SKF), and both the full and sparsified Cholesky factors, **G** and $\mathbf{G}_{\text{sparse}}$, for several maps, as well as the corresponding C-SKF mapped-feature update time (for 20 measurements) are available in Table II. As expected, the sparse Cholesky factor requires significantly less disk space than the dense covariance, and its memory footprint grows at a slower rate as the map size increases. Moreover, and as expected, we observe a decrease in map density and update time after sparsification. These gains are more pronounced in the case of maps that contain either a large number of loop-closures or extended periods of time hovering over the same scene. The smaller maps (i.e., sub-maps) allow for real-time operation (albeit at a lower frequency than the camera's 30Hz capture rate); as a point of comparison, the perfect map approximation method requires on average 7 ms to process 20 measurements to mapped-features, regardless of map size.

VII. Conclusion And Future Work

In this paper, we focused on the problem of performing approximate, but consistent map-based localization. Specifically, and motivated from the linear (in the map's size) processing cost, but quadratic memory requirements of the Schmidt-Kalman filter (SKF) when applied to map-based localization, we introduced the Cholesky (C)-SKF, which uses the map's Cholesky factor to model the information (and thus uncertainty) in the prior map. By doing so, and given the sparsity of the Cholesky factor, the C-SKF has memory requirements only linear in the map's size. Moreover, its equations are factored in such a form so as to avoid inverting the Cholesky factor of the map's Hessian. Despite the gains in memory efficiency, however, the processing cost of the C-SKF may grow more than linearly in the map's size, which motivated us to introduce two relaxations: (i) the sC-SKF, which uses the sub-maps obtained by partitioning the original map, and (ii) a Cholesky factor sparsification method that selects and maintains a subset of loop-closure measurements based on their temporal distribution. Lastly, the computational requirements of the proposed C-SKF and sC-SKF were assessed using a Google Tango tablet, where we demonstrated their superior performance against other approximate, but inconsistent, map-based approaches through real-world experiments. As future work, we seek to find an optimal partitioning of sub-maps, and exploit additional metrics for loop-closure measurement selection.

References

[1] M. Agrawal and K. Konolige, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 13–16 2007, pp. 225–234.

[3] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Trans. on Robotics*, vol. 25, no. 2, pp. 264–280, Apr. 2009.

[4] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," in *Proc. of Robotics: Science and Systems Conference*, Rome, Italy, July 13–17 2015.

[5] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt, "Scalable 6-dof localization on mobile devices," in *Proc. of the European Conference on Computer Vision*, Zurich, Switzerland, Sept. 6–12 2014, pp. 649–663.

[6] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "Global localization from monocular SLAM on a mobile phone," *IEEE Trans. on Visualization and Computer Graphics*, vol. 20, no. 4, pp. 531–539, Apr. 2014.

[7] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Observability-based rules for designing consistent EKF SLAM estimators," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, Apr. 2010.

[8] S. F. Schmidt, "Applications of state space methods to navigation problems," *Advanced Control Systems*, vol. 3, pp. 293–340, 1966.

[9] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[10] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[11] S. J. Julier, "A sparse weight Kalman filter approach to simultaneous localisation and map building," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Maui, HI, USA, Oct. 29 – Nov. 3 2001, pp. 1251–1256.

[12] E. D. Nerurkar and S. I. Roumeliotis, "Power-SLAM: a linear-complexity, anytime algorithm for SLAM," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 772–788, May 2011.

[13] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. on Robotics*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[14] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Vision Algorithms: Theory and Practice*. Springer–Verlag, 2000, pp. 298–375.

[15] J. J. Leonard and H. J. S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Proc. of the International Symposium on Robotic Research*, Snowbird, UT, Oct. 9–12 2000, pp. 169–178.

[16] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact, out-of-core, submap-based SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 10–14 2007, pp. 1678–1685.

[17] C. Guo, K. Sartipi, R. DuToit, G. Georgiou, R. Li, J. O'Leary, E. Nerurkar, J. Hesch, and S. Roumeliotis, "Large-scale cooperative 3D visual-inertial mapping in a Manhattan world," in *Proc. of the IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, May 16–21 2016, pp. 1071–1078.

[18] N. Carlevaris-Bianco and R. M. Eustice, "Conservative edge sparsification for graph SLAM node removal," in *Proc. of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 31 – June 7 2014, pp. 854–860.

[19] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear graph sparsification for SLAM," in *Proceedings of Robotics: Science and Systems*, Berkeley, CA, USA, July 12–16 2014.

[20] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6–10 2013, pp. 5748–5755.

[21] L. Paull, G. Huang, and J. J. Leonard, "A unified resource-constrained framework for graph SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, May 16–21 2016, pp. 1346–1353.

[22] E. Eade, P. Fong, and M. E. Munich, "Monocular graph SLAM with complexity reduction," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 18–22 2010, pp. 3017–3024.

[23] G. Huang, M. Kaess, and J. J. Leonard, "Consistent sparsification for graph optimization," in *Proc. of the European Conference on Mobile Robots*, Sept. 25–27 2013, pp. 150–157.

[24] K. Eckenhoff, L. Paull, and G. Huang, "Decoupled, consistent node removal and edge sparsification for graph-based SLAM," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea, Oct. 9–14 2016, pp. 3275–3282.

[25] J. Vial, H. Durrant-Whyte, and T. Bailey, "Conservative sparsification for efficient and consistent approximate estimation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, Sept. 25–30 2011, pp. 886–893.

[26] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, 2009.

[27] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, College Park, MD, June 16–21 2012, pp. 510–517.

[28] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, NY, June 17–22 2006, pp. 2161–2168.

[29] C. Guo, D. Kottas, R. DuToit, A. Ahmed, R. Li, and S. Roumeliotis, "Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps," in *Proceedings of Robotics: Science and Systems*, Berkeley, CA, USA, July 12–16 2014.

[30] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the Alvey Vision Conference*, Manchester, UK, Aug. 31 – Sept. 2 1988, pp. 147–151.

[31] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the International Joint Conference on Artificaial Intelligence*, Vancouver, British Columbia, Aug. 24–28 1981, pp. 674–679.

[32] R. C. DuToit, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "Consistent map-based 3D localization on mobile devices," *arXiv preprint arXiv:1604.08087*, 2016.

[33] F. M. Mirzaei and S. I. Roumeliotis, "A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1143–1156, Oct. 2008.

[34] A. B. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics, 1997, vol. 174.

[35] Z. Kukelova, M. Bujnak, and T. Pajdla, "Closed-form solutions to minimal absolute pose problems with known vertical direction," in *Proc. of the Asian Conference on Computer Vision*, Queenstown, New Zealand, Nov. 8–12 2011, pp. 216–229.

[36] Google, "Project Tango," https://get.google.com/tango/.