Robotics: Science and Systems 2015
Rome, Italy, July 13-17, 2015

A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices

Kejian J. Wu*, Ahmed M. Ahmed†, Georgios A. Georgiou† and Stergios I. Roumeliotis†
*Department of Electrical and Computer Engineering
†Department of Computer Science and Engineering
University of Minnesota, Minneapolis, Minnesota, USA
Email: {kejian, medhat, georgiou, stergios}@cs.umn.edu

***Abstract*—In this paper, we present a square-root inverse sliding window filter (SR-ISWF) for vision-aided inertial navigation systems (VINS). While regular inverse filters suffer from numerical issues, employing their square-root equivalent enables the usage of single-precision number representations, thus achieving considerable speed ups as compared to double-precision alternatives on resource-constrained mobile platforms. Besides a detailed description of the SR-ISWF for VINS, which focuses on the numerical procedures that enable exploiting the problem's structure for gaining in efficiency, this paper presents a thorough validation of the algorithm's processing requirements and achieved accuracy. In particular, experiments are conducted using a commercial-grade cell phone, where the proposed algorithm is shown to achieve the same level of estimation accuracy, when compared to state-of-the-art VINS algorithms, with significantly higher speed.**

I. Introduction and Related Work

Combining inertial data, from an inertial measurement unit (IMU), with visual observations, from a camera, in what is known as a vision-aided inertial navigation system (VINS), has become a popular choice for navigating in GPS-denied areas (e.g., underground, in space, or indoors). Moreover, commercial-grade mobile devices (e.g., cell phones) have recently been recognized as promising platforms for VINS, because of their sensing capabilities, and wide-spread, low-cost availability [1].

It is well known that under certain assumptions the Maximum a Posteriori (MAP) estimator for VINS can be cast as a nonlinear batch least-squares (BLS) problem, often referred to as bundle adjustment (BA) in the computer vision literature. BLS methods (e.g., [25, 5]), however, that optimize over the sensor's entire trajectory and the map of the environment, have processing and memory requirements that increase at least linearly (often quadratically) with time, and thus are not amenable to real-time implementations. Although various approximations of the BLS have been proposed (e.g., [11, 12, 22]) to reduce the processing cost, they are still far from being able to generate real-time estimates on resource-constrained mobile devices.

To achieve constant processing time, filtering approaches or fixed-lag smoothers (FLS) optimize over a *bounded-size* sliding window of recent states by marginalizing out past states and measurements. Popular filtering techniques for VINS are either based on the extended Kalman filter (EKF) (e.g., [20, 10, 8]) or the inverse filter (INVF) (e.g., [23, 6, 15]). While real-time performance on mobile devices has been successfully demonstrated using EKFs (e.g., [17, 8]), to the best of our knowledge, no real-time INVF-based system has been shown on resource-constrained platforms such as cell phones.

Despite the fact that the INVF is mathematically equivalent to the EKF, one of the main disadvantages of the INVF is its poor numerical properties in general [19], but primarily for VINS. Specifically, the Hessian matrix employed by the INVF is typically ill-conditioned (i.e., condition number $\sim 10^9$ or higher), which necessitates using double-precision arithmetic, else the numerical errors can easily become the dominant error source affecting estimation accuracy, or even cause the filter to diverge. Achieving efficient VINS solutions on mobile devices, however, requires single-precision arithmetic for the following reasons: i) A 32-bit single-precision representation occupies less hardware resources, and the corresponding operations are likely to be faster, than 64-bit double-precision representations, and ii) most current mobile devices feature ARM NEON co-processors that provide a 4-fold processing speed increase when using 32-bit number representations.

In order to overcome the numerical limitations of the INVF, square-root inverse filters [19, 2] have been developed in the past that improve numerical stability by maintaining and updating the square root of the Hessian matrix. This way, square root filters can yield twice the effective precision of regular filters when using the same wordlength.[1] Recently, a sliding window inverse smoother for VINS, based on iSAM2 [12], was proposed in [4]. Few details, however, were provided for the realization of this algorithm, or its processing requirements. In contrast, in this paper we focus on taking advantage of the structure of the VINS problem when formulated as an inverse square root filter to achieve computational gains that allow operating in real time on mobile devices.

Specifically, in this paper, we present the square root inverse sliding window filter (SR-ISWF) for VINS, which maintains the upper triangular Cholesky factor of the Hessian matrix of a sliding window of recent states. Our main contributions are:

- To the best of our knowledge, we present the first VINS in the information domain that operates in real time on

[1]Note that the condition number of the square root matrix is the square root of the condition number of the corresponding Hessian matrix.

resource-constrained devices, specifically, cell phones.
- Our implementation of the proposed algorithm achieves more than double the speed, with comparable accuracy, of the multi-state constraint Kalman filter (MSC-KF) [20], which has been successfully demonstrated on mobile devices. Similar results are achieved when compared with the short-term smoother in [4].
- The proposed SR-ISWF for VINS handles limitations of low-cost devices, by estimating on-the-fly the IMU-camera extrinsic calibration parameters, time-synchronization, and the camera's rolling shutter time.

The rest of this paper is structured as follows: In Section II, we briefly overview the key components of a VINS. Section III presents the derivations of the proposed SR-ISWF algorithm that leads to efficient implementations. Experimental results over several cell phone datasets are shown in Section IV, comparing the performance of the proposed SR-ISWF against state-of-the-art VINS. Finally, Section V concludes the paper.

II. Vision-aided Inertial Navigation System (VINS)

In this section, we provide a brief description of the key components of the proposed VINS.

A. System State

At each time step k, the system maintains the following state vector

$$\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_S^T & \mathbf{x}_{C_{k-M+1}}^T & \cdots & \mathbf{x}_{C_k}^T & \mathbf{x}_P^T & \mathbf{x}_{E_k}^T\end{bmatrix}^T \quad (1)$$

where $\mathbf{x}_S$ is the state vector of the current SLAM features under estimation, with $\mathbf{x}_S = \begin{bmatrix}{}^G\mathbf{p}_{f_1}^T & \cdots & {}^G\mathbf{p}_{f_n}^T\end{bmatrix}^T$, and ${}^G\mathbf{p}_{f_j}$, for $j = 1, \cdots, n$, denotes the Euclidean coordinates of the point feature $\mathbf{f}_j$ in the global frame $\{G\}$. $\mathbf{x}_{C_i}$, for $i = k + M - 1, \cdots, k$, represents the state vector of the cloned[2] IMU poses at time step i, where M is the size of the sliding window. Each cloned state is defined as

$$\mathbf{x}_{C_i} = \begin{bmatrix}{}^{I_i}\mathbf{q}_G^T & {}^G\mathbf{p}_{I_i}^T & t_{d_i}\end{bmatrix}^T \quad (2)$$

where ${}^{I_i}\mathbf{q}_G$ is the quaternion representation of the orientation of the global frame $\{G\}$ in the IMU's frame of reference $\{I_i\}$, ${}^G\mathbf{p}_{I_i}$ is the position of $\{I_i\}$ in $\{G\}$, and t_{d_i} is the IMU-camera time offset,[3] at time step i, similar to the definition in [8]. The parameter state vector, $\mathbf{x}_P$, consists of the constant parameters:

$$\mathbf{x}_P = \begin{bmatrix}{}^I\mathbf{q}_C^T & {}^I\mathbf{p}_C^T & t_r\end{bmatrix}^T \quad (3)$$

where ${}^I\mathbf{q}_C$ is the quaternion representation of the orientation of the camera frame $\{C\}$ in the IMU's frame of reference $\{I\}$, ${}^I\mathbf{p}_C$ is the position of $\{C\}$ in $\{I\}$, and t_r is the rolling-shutter time of the camera (i.e., the readout time of each image, which is constant). Finally, the states necessary for modelling the IMU biases and determining the sensor's current speed are kept in

$$\mathbf{x}_{E_k} = \begin{bmatrix}\mathbf{b}_{g_k}^T & {}^G\mathbf{v}_{I_k}^T & \mathbf{b}_{a_k}^T\end{bmatrix}^T \quad (4)$$

where $\mathbf{b}_{g_k}$ and $\mathbf{b}_{a_k}$ correspond to the gyroscope and accelerometer biases, respectively, and ${}^G\mathbf{v}_{I_k}$ is the velocity of $\{I\}$ in $\{G\}$, at time step k.

B. Inertial Measurement Equations and Corresponding Cost Terms

The IMU provides measurements of the device's rotational velocity and linear acceleration contaminated by white Gaussian noise and time-varying biases. The biases are modelled as random walks driven by white, zero-mean Gaussian noise processes $\mathbf{n}_{wg}(t)$ and $\mathbf{n}_{wa}(t)$ [see (6)], while the gyroscope and accelerometer measurements, $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ are:

$$\begin{aligned}\boldsymbol{\omega}_m(t) &= {}^I\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t)\\ \mathbf{a}_m(t) &= \mathbf{C}({}^I\mathbf{q}_G(t))({}^G\mathbf{a}(t) - {}^G\mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a(t)\end{aligned} \quad (5)$$

where the noise terms, $\mathbf{n}_g(t)$ and $\mathbf{n}_a(t)$ are modelled as zero-mean, white Gaussian noise processes, while the gravitational acceleration ${}^G\mathbf{g}$ is considered a known deterministic constant. The device's rotational velocity ${}^I\boldsymbol{\omega}(t)$ and linear acceleration ${}^G\mathbf{a}(t)$, in (5), can be used to relate consecutive IMU poses through the device's continuous-time system equations:

$$\begin{aligned}&{}^I\dot{\mathbf{q}}_G(t) = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t) - \mathbf{b}_g(t) - \mathbf{n}_g(t)){}^I\mathbf{q}_G(t)\\ &{}^G\dot{\mathbf{v}}_I(t) = \mathbf{C}({}^I\mathbf{q}_G(t))^T(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)) + {}^G\mathbf{g}\\ &{}^G\dot{\mathbf{p}}_I(t) = {}^G\mathbf{v}_I(t) \qquad \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \qquad \dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t)\\ &{}^I\dot{\mathbf{q}}_C(t) = 0 \quad {}^I\dot{\mathbf{p}}_C(t) = 0 \quad \dot{t}_d(t) = \mathbf{n}_{td}(t) \quad \dot{t}_r(t) = 0\end{aligned} \quad (6)$$

where, $\boldsymbol{\Omega}(\boldsymbol{\omega}) \triangleq \begin{bmatrix}-\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega}\\ -\boldsymbol{\omega}^T & 0\end{bmatrix}$ for $\boldsymbol{\omega} \in \mathbb{R}^3$, and the IMU-camera time offset t_d is modelled as random walk driven by the white zero-mean Gaussian noise process $\mathbf{n}_{td}$.

Given the inertial measurements $\mathbf{u}_{k,k+1} = \begin{bmatrix}\boldsymbol{\omega}_{m_k}^T & \mathbf{a}_{m_k}^T\end{bmatrix}^T$, analytical integration of (6) within the time interval $\begin{bmatrix}t_k, & t_{k+1}\end{bmatrix}$ is used to determine the discrete-time system equations [24], which imposes a constraint between the consecutive states $\mathbf{x}_{I_k}$ and $\mathbf{x}_{I_{k+1}}$ as:

$$\mathbf{x}_{I_{k+1}} = \mathbf{f}(\mathbf{x}_{I_k},\ \mathbf{u}_{k,k+1} - \mathbf{w}_{k,k+1}) \quad (7)$$

where $\mathbf{x}_{I_k} = \begin{bmatrix}\mathbf{x}_{C_k}^T & \mathbf{x}_{E_k}^T\end{bmatrix}^T$, and $\mathbf{w}_{k,k+1}$ is the discrete-time zero-mean white Gaussian noise affecting the IMU measurements with covariance $\mathbf{Q}_k$.

Linearizing (7), around the state estimates $\hat{\mathbf{x}}_{I_k}$ and $\hat{\mathbf{x}}_{I_{k+1}}$, results in the following IMU measurement model, relating the error states $\widetilde{\mathbf{x}}_{I_k}$ and $\widetilde{\mathbf{x}}_{I_{k+1}}$:

$$\begin{aligned}\widetilde{\mathbf{x}}_{I_{k+1}} &= (\mathbf{f}(\hat{\mathbf{x}}_{I_k}, \mathbf{u}_{k,k+1}) - \hat{\mathbf{x}}_{I_{k+1}})\\ &\quad + \boldsymbol{\Phi}_{k+1,k}\widetilde{\mathbf{x}}_{I_k} + \mathbf{G}_{k+1,k}\mathbf{w}_{k,k+1}\end{aligned} \quad (8)$$

where $\boldsymbol{\Phi}_{k+1,k}$ and $\boldsymbol{G}_{k+1,k}$ are the corresponding Jacobians. In this paper, we define the error state $\widetilde{\mathbf{x}}$ as the difference between the true state $\mathbf{x}$ and the state estimate $\hat{\mathbf{x}}$ employed for linearization (i.e., $\widetilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$), while for the quaternion $\mathbf{q}$ we employ

[2] We refer to the same stochastic cloning as in the MSCKF [20], for maintaining past IMU poses in a sliding window estimator.

[3] The IMU and camera operate at different frequencies, and the time stamps reported by the camera are not accurate. Thus, the unknown time difference between the two sensors is time-varying and needs to be estimated per image.

a multiplicative error model $\widetilde{\mathbf{q}} = \mathbf{q} \otimes \hat{\mathbf{q}}^{-1} \simeq \begin{bmatrix}\frac{1}{2}\delta\boldsymbol{\theta}^T & 1\end{bmatrix}^T$, where $\delta\boldsymbol{\theta}$ is a minimal representation of the attitude error.

Hence, the inertial measurements $\mathbf{u}_{k,k+1}$ contribute a linearized cost term of the form:

$$\mathcal{C}_u(\widetilde{\mathbf{x}}_{I_k}, \widetilde{\mathbf{x}}_{I_{k+1}}) = || \begin{bmatrix}\boldsymbol{\Phi}_{k+1,k} & -\mathbf{I}\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{x}}_{I_k} \\ \widetilde{\mathbf{x}}_{I_{k+1}}\end{bmatrix} - (\hat{\mathbf{x}}_{I_{k+1}} - \mathbf{f}(\hat{\mathbf{x}}_{I_k}, \mathbf{u}_{k,k+1}))||^2_{\mathbf{Q}'_k} \quad (9)$$

where $\mathbf{Q}'_k = \mathbf{G}_{k+1,k}\mathbf{Q}_k\mathbf{G}^T_{k+1,k}$.

C. Visual Measurement Equations and Corresponding Cost Terms

We use point features extracted from consecutive images as visual measurements to be processed by the estimator. When working with commercial-grade mobile devices (e.g., cell phones), the images suffer from the rolling-shutter effect; that is the image pixel rows are read sequentially in time, so each row has a different actual observation time. In addition, there exists a time-varying offset between the IMU and the camera's time stamps (i.e., the two clocks are not synchronized). To handle these issues, we use the interpolation model of [8], where each camera pose is interpolated using the two closest cloned poses. In particular, the measurement model is:

$$\mathbf{z}^j_k = \mathbf{h}({}^{C_{k+t}}\mathbf{p}_{f_j}) + \mathbf{n}^j_k = \pi({}^{C_{k+t}}\mathbf{p}_{f_j}) + \mathbf{n}^j_k \quad (10)$$

with $\pi(\begin{bmatrix}x & y & z\end{bmatrix}^T) = \begin{bmatrix}\frac{x}{z} & \frac{y}{z}\end{bmatrix}^T$, where ${}^{C_{k+t}}\mathbf{p}_{f_j}$ is the feature position expressed in the camera frame of reference at the exact image-acquisition time instant, $\mathbf{n}^j_k$ is zero-mean, white Gaussian noise with covariance $\sigma^2\mathbf{I}_2$, and $\mathbf{I}_2$ is the 2×2 identity matrix. Note that without loss of generality, we express the feature measurement (10) in normalized pixel coordinates, after we perform intrinsic camera calibration offline [3]. Linearizing (10) around current state estimates (where the feature positions are obtained through triangulation using all the measurements in the current window) yields:

$$\widetilde{\mathbf{z}}^j_k = \mathbf{H}^j_{x,k}\,\widetilde{\mathbf{x}}_F + \mathbf{H}^j_{f,k}\,{}^G\widetilde{\mathbf{p}}_{f_j} + \mathbf{n}^j_k \quad (11)$$

where $\mathbf{H}^j_{x,k}$ and $\mathbf{H}^j_{f,k}$ are the corresponding Jacobians evaluated at the state estimate $\hat{\mathbf{x}}_k$, and we define

$$\mathbf{x}_F = \begin{bmatrix}\mathbf{x}^T_{C_{k-M+1}} & \cdots & \mathbf{x}^T_{C_k} & \mathbf{x}^T_P & \mathbf{x}^T_{E_k}\end{bmatrix}^T \quad (12)$$

which is the state vector comprising all current states except the SLAM features. Stacking together all N_j observations to this feature, we get:

$$\widetilde{\mathbf{z}}^j = \mathbf{H}^j_x\widetilde{\mathbf{x}}_F + \mathbf{H}^j_f\,{}^G\widetilde{\mathbf{p}}_{f_j} + \mathbf{n}^j \quad (13)$$

Consider a square orthonormal matrix $\mathbf{Q}_j$, partitioned as $\mathbf{Q}_j = \begin{bmatrix}\mathbf{S}_j & \mathbf{U}_j\end{bmatrix}$, where the 3 columns of $\mathbf{S}_j$ span the column space of $\mathbf{H}^j_f$, while the $2N_j - 3$ rows of $\mathbf{U}_j$, its left nullspace. Then, the visual measurements $\mathbf{z}^j$ contribute a linearized cost term:

$$\begin{aligned}\mathcal{C}_z(\widetilde{\mathbf{x}}_F, {}^G\widetilde{\mathbf{p}}_{f_j}) &= ||\mathbf{H}^j_x\widetilde{\mathbf{x}}_F + \mathbf{H}^j_f\,{}^G\widetilde{\mathbf{p}}_{f_j} - \widetilde{\mathbf{z}}^j||^2_{\sigma^2\mathbf{I}_{2N_j}} \quad (14)\\ &= ||\mathbf{Q}^T_j\mathbf{H}^j_x\widetilde{\mathbf{x}}_F + \mathbf{Q}^T_j\mathbf{H}^j_f\,{}^G\widetilde{\mathbf{p}}_{f_j} - \mathbf{Q}^T_j\widetilde{\mathbf{z}}^j||^2_{\sigma^2\mathbf{I}_{2N_j}}\\ &= ||\mathbf{S}^T_j\mathbf{H}^j_x\widetilde{\mathbf{x}}_F + \mathbf{S}^T_j\mathbf{H}^j_f\,{}^G\widetilde{\mathbf{p}}_{f_j} - \mathbf{S}^T_j\widetilde{\mathbf{z}}^j||^2_{\sigma^2\mathbf{I}_3}\\ &\quad + ||\mathbf{U}^T_j\mathbf{H}^j_x\widetilde{\mathbf{x}}_F - \mathbf{U}^T_j\widetilde{\mathbf{z}}^j||^2_{\sigma^2\mathbf{I}_{2N_j-3}}\\ &= ||\mathbf{F}^j_1\widetilde{\mathbf{x}}_F + \mathbf{R}^j_f\,{}^G\widetilde{\mathbf{p}}_{f_j} - \widetilde{\mathbf{z}}^j_1||^2_{\sigma^2\mathbf{I}_3} \quad (15)\\ &\quad + ||\mathbf{F}^j_2\widetilde{\mathbf{x}}_F - \widetilde{\mathbf{z}}^j_2||^2_{\sigma^2\mathbf{I}_{2N_j-3}} \quad (16)\\ &= \mathcal{C}_{z^1}(\widetilde{\mathbf{x}}_F, {}^G\widetilde{\mathbf{p}}_{f_j}) + \mathcal{C}_{z^2}(\widetilde{\mathbf{x}}_F) \quad (17)\end{aligned}$$

with

$$\mathbf{F}^j_1 \triangleq \mathbf{S}^T_j\mathbf{H}^j_x \quad \mathbf{F}^j_2 \triangleq \mathbf{U}^T_j\mathbf{H}^j_x \quad \mathbf{R}^j_f \triangleq \mathbf{S}^T_j\mathbf{H}^j_f$$
$$\widetilde{\mathbf{z}}^j_1 \triangleq \mathbf{S}^T_j\widetilde{\mathbf{z}}^j \quad \widetilde{\mathbf{z}}^j_2 \triangleq \mathbf{U}^T_j\widetilde{\mathbf{z}}^j \quad (18)$$

To perform the above orthonormal transformation, i.e., to compute (18), we apply Givens rotations [7] (pages 252-253) to triangularize $\mathbf{H}^j_f$, while the other quantities are obtained *in place* through the Givens process. As a result of this Givens, in (18), the Jacobian $\mathbf{R}^j_f$ is square and upper-triangular, while $\mathbf{F}^j_2$ is block upper-triangular. These structures will lead us to efficient update steps later on.

D. Visual-Information Management

Our VINS employs two types of visual measurements, as in [20, 21, 16], so as to provide high estimation accuracy while remaining computationally efficient. Hence, we designed an information management module, which classifies the observed features, so as to be appropriately processed, into two categories:

- **SLAM features**: These features are added in the state vector (1) and updated across time. In terms of the cost function in (17), $\mathcal{C}_{z^1}$ is used for initializing new SLAM features, while $\mathcal{C}_{z^2}$ is exploited for state update.
  By maintaining a structure of the scene, SLAM features increase the estimation accuracy and improve the estimator's robustness, especially when viewed over many frames. This, however, comes at higher cost as compared to the MSCKF features (described later on). Hence, the information manager selects as new SLAM features those whose tracks span the entire sliding window (since these tracks are likely to last longer), while limits the number of SLAM features in the estimator. Once a SLAM feature is successfully initialized by the filter, it stays in the VINS state vector until the track is lost (i.e., when this feature is no longer observed by the newest pose in the sliding window); then this feature will be marginalized and removed from the state vector. Note that this is still a sliding window scheme though SLAM features can stay longer than the window-size duration dictates.
- **MSCKF features**: These features are processed as in the MSC-KF approach [20], hence the name,[4] where the feature states are marginalized from the measurement equation (13) to generate constraints between poses. Compared with SLAM, this approach directly exploits the cost term $\mathcal{C}_{z^2}$ in (17). Note, however, that all the

[4]Throughout this paper, we use the term "MSC-KF" to denote the algorithm itself in [20], while "MSCKF features" signifies the features processed in such a manner.

information on the poses from the measurements is absorbed into $\mathcal{C}_{z^2}$, since $\mathcal{C}_{z^1}$ contains only information about the feature's position.

MSCKF feature measurements provide "local" information relating multiple poses in each sliding window. They require less computations than SLAM since their feature states are not directly estimated. After the SLAM features have been selected, the information manager classifies all the remaining feature tracks into the MSCKF category.

Finally, and in order to address the processing limitations of resource-constrained platforms, the information manager trades estimation accuracy for computational efficiency, by limiting the number of features processed in each update. Moreover, MSCKF features are selected based on their track lengths, with higher priority given to the longer tracks.

III. Estimation Algorithm Description

In this section, we describe in detail the main steps of the SR-ISWF algorithm. Our presentation will be from the cost function perspective: We first show the effect that each step has on the cost function being minimized, and then present the corresponding equations, with special attention to specific problem structures for efficient implementation.

At each time step k, our objective is to minimize the cost function $\mathcal{C}_k^{\oplus}$ that contains all the information available so far:

$$\begin{aligned}\mathcal{C}_k^{\oplus} &= \mathcal{C}_{k-1} + \mathcal{C}_u + \mathcal{C}_{\mathbb{Z}_R} + \mathcal{C}_{\mathbb{Z}_S} + \mathcal{C}_{\mathbb{Z}_M} \\ &= \mathcal{C}_{k-1} + \mathcal{C}_u + \mathcal{C}_{\mathbb{Z}_R} + \mathcal{C}_{\mathbb{Z}_S^1} + \mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2} \end{aligned} \tag{19}$$

where $\mathcal{C}_u$ represents the cost term arising from the IMU measurement $\mathbf{u}_{k-1,k}$, $\mathcal{C}_{\mathbb{Z}_R}$ from the visual re-observations of the active SLAM features, $\mathcal{C}_{\mathbb{Z}_S}$ from the camera measurements to new SLAM features (to be initialized), and $\mathcal{C}_{\mathbb{Z}_M}$ from the visual measurements to the MSCKF features. The new-SLAM cost term is further split into $\mathcal{C}_{\mathbb{Z}_S^1}$ and $\mathcal{C}_{\mathbb{Z}_S^2}$, while the MSCKF cost term only consists of $\mathcal{C}_{\mathbb{Z}_M^2}$, according to (17) (see Sect. II-D). Finally, all the prior information obtained from the previous time step is contained in

$$\mathcal{C}_{k-1}(\widetilde{\mathbf{x}}_{k-1}) = \|\mathbf{R}_{k-1}\widetilde{\mathbf{x}}_{k-1} - \mathbf{r}_{k-1}\|^2 \tag{20}$$

where $\|\cdot\|$ denotes the standard vector 2-norm, $\mathbf{R}_{k-1}$ and $\mathbf{r}_{k-1}$ are the prior (upper-triangular) information *factor* matrix (i.e., the square root of the Hessian) and residual vector, respectively, and $\widetilde{\mathbf{x}}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}$ the error state from time step $k-1$ [see (1)]. Note that in (20), $\mathbf{r}_{k-1} = \mathbf{0}$ and will be updated along with the information factor $\mathbf{R}_{k-1}$.

We hereafter describe how the SR-ISWF algorithm combines each cost term in (19) to eventually obtain $\mathcal{C}_k^{\oplus}$. Meanwhile, we will show how the system's state vector evolves from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$. An overview of the SR-ISWF algorithm is shown in Algorithm 1.

A. Propagation

In our sliding window, a new pose state $\mathbf{x}_{I_k}$ [see (7)] is added to the current state vector at each time step k:

$$\mathbf{x}_k^{\ominus} = \begin{bmatrix}\mathbf{x}_{k-1}^T & \mathbf{x}_{I_k}^T\end{bmatrix}^T \tag{21}$$

Algorithm 1 SR-ISWF

Input:

- Prior estimate $\hat{\mathbf{x}}_{k-1}$
- Upper-triangular prior information factor $\mathbf{R}_{k-1}$
- Inertial measurements $\mathbf{u}_{k-1,k}$
- SLAM re-observation measurements $\mathbb{Z}_R$
- New SLAM feature measurements $\mathbb{Z}_S$
- MSCKF feature measurements $\mathbb{Z}_M$

Function Steps:

Propagation [see (23)]
Marginalization [see (28)]
Covariance factor recovery [see (33)]
Information factor update:

- SLAM re-observations [see (37)]
- New SLAM features initialization [see (45)]
- New SLAM & MSCKF pose constraints [see (50) and (48)]

State update [see (53) and (54)]

using the IMU measurement $\mathbf{u}_{k-1,k}$. Hence the cost function, which initially comprised only the prior $\mathcal{C}_{k-1}$, becomes

$$\begin{aligned}\mathcal{C}_k^{\ominus}(\widetilde{\mathbf{x}}_k^{\ominus}) &= \mathcal{C}_{k-1}(\widetilde{\mathbf{x}}_{k-1}) + \mathcal{C}_u(\widetilde{\mathbf{x}}_{I_{k-1}}, \widetilde{\mathbf{x}}_{I_k}) \\ &= \|\mathbf{R}_{k-1}\widetilde{\mathbf{x}}_{k-1} - \mathbf{r}_{k-1}\|^2 + \| \begin{bmatrix}\boldsymbol{\Phi}_{k,k-1} & -\mathbf{I}\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{x}}_{I_{k-1}} \\ \widetilde{\mathbf{x}}_{I_k}\end{bmatrix} \\ &\quad - (\hat{\mathbf{x}}_{I_k} - \mathbf{f}(\hat{\mathbf{x}}_{I_{k-1}}, \mathbf{u}_{k-1,k}))\|^2_{\mathbf{Q}'_k} \\ &= \|\mathbf{R}_k^{\ominus}\widetilde{\mathbf{x}}_k^{\ominus} - \mathbf{r}_k^{\ominus}\|^2 \end{aligned} \tag{22}$$

from (20) and (9), with

$$\begin{aligned}\mathbf{R}_k^{\ominus} &= \begin{bmatrix}\mathbf{R}_{k-1} & \mathbf{0} \\ \mathbf{V}_1 & \mathbf{V}_2\end{bmatrix}, \quad \mathbf{V}_2 = -\mathbf{Q}_k'^{-\frac{1}{2}} \\ \mathbf{V}_1 &= \begin{bmatrix}\mathbf{0}\cdots\mathbf{0} & \mathbf{Q}_k'^{-\frac{1}{2}}\boldsymbol{\Phi}_{k,k-1}^{(C)} & \mathbf{0}\cdots\mathbf{0} & \mathbf{Q}_k'^{-\frac{1}{2}}\boldsymbol{\Phi}_{k,k-1}^{(E)}\end{bmatrix} \\ \mathbf{r}_k^{\ominus} &= \begin{bmatrix}\mathbf{r}_{k-1} \\ \mathbf{Q}_k'^{-\frac{1}{2}}(\hat{\mathbf{x}}_{I_k} - \mathbf{f}(\hat{\mathbf{x}}_{I_{k-1}}, \mathbf{u}_{k-1,k}))\end{bmatrix} \end{aligned} \tag{23}$$

where $\boldsymbol{\Phi}_{k,k-1}^{(C)}$ and $\boldsymbol{\Phi}_{k,k-1}^{(E)}$ are block columns of the Jacobian $\boldsymbol{\Phi}_{k,k-1}$ with respect to the clone and extra IMU states [see (2) and (4)], respectively. And here $\mathbf{r}_k^{\ominus} = \mathbf{0}$, since $\mathbf{r}_{k-1} = \mathbf{0}$ and $\hat{\mathbf{x}}_{I_k} = \mathbf{f}(\hat{\mathbf{x}}_{I_{k-1}}, \mathbf{u}_{k-1,k})$ from state propagation (7). Note that the resulting factor $\mathbf{R}_k^{\ominus}$ is not upper-triangular in (23), but will be triangularized in the next step.

B. Marginalization

To maintain constant computational complexity, at time step k, the SR-ISWF marginalizes the following states: Past SLAM features, $\widetilde{\mathbf{x}}_{DS}$, whose tracks are lost (DS for "disappeared SLAM"), the oldest clone $\widetilde{\mathbf{x}}_{C_{k-M}}$, and the extra IMU states $\widetilde{\mathbf{x}}_{E_{k-1}}$ from the previous time step. If we define the (error)

state vector consisting of all states to be marginalized as

$$\widetilde{\mathbf{x}}_k^M = \begin{bmatrix}\widetilde{\mathbf{x}}_{DS}^T & \widetilde{\mathbf{x}}_{C_{k-M}}^T & \widetilde{\mathbf{x}}_{E_{k-1}}^T\end{bmatrix}^T \tag{24}$$

and denote $\widetilde{\mathbf{x}}_k^R$ the remaining states [following the structure in (1)] after removing $\widetilde{\mathbf{x}}_k^M$ from $\widetilde{\mathbf{x}}_k^\ominus$, then we can write

$$\mathbf{P}_M \widetilde{\mathbf{x}}_k^\ominus = \begin{bmatrix}\widetilde{\mathbf{x}}_k^{M^T} & \widetilde{\mathbf{x}}_k^{R^T}\end{bmatrix}^T \tag{25}$$

where $\mathbf{P}_M$ is a permutation matrix. In terms of the cost function, marginalization corresponds to removing $\widetilde{\mathbf{x}}_k^M$ from the cost function by minimizing with respect to it, i.e.,

$$\mathcal{C}_k^M(\widetilde{\mathbf{x}}_k^R) = \min_{\widetilde{\mathbf{x}}_k^M} \mathcal{C}_k^\ominus(\widetilde{\mathbf{x}}_k^\ominus) = \min_{\widetilde{\mathbf{x}}_k^M} \mathcal{C}_k^\ominus(\widetilde{\mathbf{x}}_k^M, \widetilde{\mathbf{x}}_k^R) \tag{26}$$

Combining (22) and (25), we obtain:

$$\begin{aligned}\mathcal{C}_k^\ominus(\widetilde{\mathbf{x}}_k^M, \widetilde{\mathbf{x}}_k^R) &= \|\mathbf{R}_k^\ominus \widetilde{\mathbf{x}}_k^\ominus - \mathbf{r}_k^\ominus\|^2 \\ &= \|\mathbf{R}_k^\ominus \mathbf{P}_M^T \mathbf{P}_M \widetilde{\mathbf{x}}_k^\ominus - \mathbf{r}_k^\ominus\|^2 \\ &= \|\mathbf{R}_k^\ominus \mathbf{P}_M^T \begin{bmatrix}\widetilde{\mathbf{x}}_k^M \\ \widetilde{\mathbf{x}}_k^R\end{bmatrix} - \mathbf{r}_k^\ominus\|^2 \end{aligned} \tag{27}$$

After performing QR factorization [7] (pages 248-250) on the column-permuted factor matrix $\mathbf{R}_k^\ominus \mathbf{P}_M^T$

$$\mathbf{R}_k^\ominus \mathbf{P}_M^T = \begin{bmatrix}\mathbf{Q}^M & \mathbf{Q}^R\end{bmatrix} \begin{bmatrix}\mathbf{R}_k^M & \mathbf{R}_k^{MR} \\ 0 & \mathbf{R}_k^R\end{bmatrix}, \tag{28}$$

(27) is written as

$$\begin{aligned}\mathcal{C}_k^\ominus(\widetilde{\mathbf{x}}_k^M, \widetilde{\mathbf{x}}_k^R) &= \left\| \begin{bmatrix}\mathbf{R}_k^M & \mathbf{R}_k^{MR} \\ 0 & \mathbf{R}_k^R\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{x}}_k^M \\ \widetilde{\mathbf{x}}_k^R\end{bmatrix} - \begin{bmatrix}\mathbf{Q}^{M^T} \\ \mathbf{Q}^{R^T}\end{bmatrix} \mathbf{r}_k^\ominus \right\|^2 \\ &= \left\| \mathbf{R}_k^M \widetilde{\mathbf{x}}_k^M + \mathbf{R}_k^{MR} \widetilde{\mathbf{x}}_k^R - \mathbf{Q}^{M^T} \mathbf{r}_k^\ominus \right\|^2 \\ &\quad + \left\| \mathbf{R}_k^R \widetilde{\mathbf{x}}_k^R - \mathbf{Q}^{R^T} \mathbf{r}_k^\ominus \right\|^2 \end{aligned} \tag{29}$$

Since $\mathbf{R}_k^M$ is invertible (from the QR process), for any $\widetilde{\mathbf{x}}_k^R$, there always exists an $\widetilde{\mathbf{x}}_k^M$ that makes the first cost term in (29) zero. Hence, combining (26) and (29), the cost function after marginalization becomes

$$\mathcal{C}_k^M(\widetilde{\mathbf{x}}_k^R) = \min_{\widetilde{\mathbf{x}}_k^M} \mathcal{C}_k^\ominus(\widetilde{\mathbf{x}}_k^M, \widetilde{\mathbf{x}}_k^R) = \left\| \mathbf{R}_k^R \widetilde{\mathbf{x}}_k^R - \mathbf{r}_k^R \right\|^2 \tag{30}$$

where $\mathbf{r}_k^R = \mathbf{Q}^{R^T} \mathbf{r}_k^\ominus = \mathbf{0}$ since $\mathbf{r}_k^\ominus = \mathbf{0}$.

C. Covariance Factor Recovery

Before employing visual measurement updates, a robust VINS system requires an outlier rejection module. In addition to the 2-Point RANSAC [13], our SR-ISWF employs the standard Mahalanobis distance test:

$$\gamma = \widetilde{\mathbf{z}}^T \mathbf{S}^{-1} \widetilde{\mathbf{z}}, \quad \mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \sigma^2 \mathbf{I} \tag{31}$$

where γ, $\widetilde{\mathbf{z}}$, $\mathbf{S}$, $\mathbf{H}$, $\mathbf{P}$, σ represent the Mahalanobis distance, measurement residual, residual covariance, measurement Jacobian, covariance matrix, and measurement noise standard deviation, respectively, for any measurement $\mathbf{z}$. Among these quantities, the only one not available is the covariance matrix $\mathbf{P}$, which when expressed as the inverse of the Hessian matrix, equals:

$$\mathbf{P} = (\mathbf{R}_k^{R^T} \mathbf{R}_k^R)^{-1} = \mathbf{R}_k^{R^{-1}} \mathbf{R}_k^{R^{-T}} = \mathbf{U}_p \mathbf{U}_p^T \tag{32}$$

$$\mathbf{U}_p = \mathbf{R}_k^{R^{-1}} \tag{33}$$

with $\mathbf{U}_p$ upper-triangular. Hence, $\mathbf{S}$ can be expressed as

$$\mathbf{S} = \mathbf{B}\mathbf{B}^T + \sigma^2 \mathbf{I}, \quad \mathbf{B} = \mathbf{H}\mathbf{U}_p \tag{34}$$

Note that this way we need not compute explicitly the covariance matrix $\mathbf{P}$, which is numerically unstable since it shares the same condition number as the Hessian.

D. Update: SLAM Re-observations

Re-observations $\mathbb{Z}_R$ of existing SLAM features are used to perform updates. Specifically, all such measurements contribute a cost term $\mathcal{C}_{\mathbb{Z}_R}$ as in (19), and thus [see (30) and (14)] the cost function becomes:

$$\begin{aligned}\mathcal{C}_k^{SR}(\widetilde{\mathbf{x}}_k^R) &= \mathcal{C}_k^M(\widetilde{\mathbf{x}}_k^R) + \mathcal{C}_{\mathbb{Z}_R}(\widetilde{\mathbf{x}}_k^R) \\ &= \|\mathbf{R}_k^R \widetilde{\mathbf{x}}_k^R - \mathbf{r}_k^R\|^2 + \|\mathbf{H}_{SR} \widetilde{\mathbf{x}}_k^R - \widetilde{\mathbf{z}}_R\|_{\sigma^2 \mathbf{I}}^2 \\ &= \left\| \begin{bmatrix}\mathbf{R}_k^R \\ \frac{1}{\sigma}\mathbf{H}_{SR}\end{bmatrix} \widetilde{\mathbf{x}}_k^R - \begin{bmatrix}\mathbf{r}_k^R \\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_R\end{bmatrix} \right\|^2 \end{aligned} \tag{35}$$

where

$$\mathbf{H}_{SR} = \begin{bmatrix}\vdots \\ \mathbf{H}_{SR}^j \\ \vdots\end{bmatrix}, \quad \widetilde{\mathbf{z}}_R = \begin{bmatrix}\vdots \\ \widetilde{\mathbf{z}}_R^j \\ \vdots\end{bmatrix}$$

$$\mathbf{H}_{SR}^j = \begin{bmatrix}\mathbf{0} \cdots \mathbf{0} & \mathbf{H}_f^j & \mathbf{0} \cdots \mathbf{0} & \mathbf{H}_x^j & \mathbf{0} \cdots \mathbf{0}\end{bmatrix} \tag{36}$$

for $j = 1, ..., N_{SR}$, with N_{SR} the total number of SLAM re-observation measurements, and the Jacobians $\mathbf{H}_f^j$, $\mathbf{H}_x^j$ and the residual $\widetilde{\mathbf{z}}_R^j$ are as in (13). If we perform the following thin QR factorization [7] (page 248):

$$\begin{bmatrix}\mathbf{R}_k^R \\ \frac{1}{\sigma}\mathbf{H}_{SR}\end{bmatrix} = \mathbf{Q}^{SR} \mathbf{R}^{SR} \tag{37}$$

Then, from (35), and after dropping a constant term, the cost function after the SLAM re-observation update becomes

$$\mathcal{C}_k^{SR}(\widetilde{\mathbf{x}}_k^R) = \|\mathbf{R}_k^{SR} \widetilde{\mathbf{x}}_k^R - \mathbf{r}_k^{SR}\|^2 \tag{38}$$

with

$$\mathbf{r}_k^{SR} = \mathbf{Q}^{SR^T} \begin{bmatrix}\mathbf{r}_k^R \\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_R\end{bmatrix} \tag{39}$$

The QR factorization in (37) is carried out very efficiently by taking advantage of the fact that $\mathbf{R}_k^R$ is upper-triangular. Furthermore, $\mathbf{r}_k^{SR}$ is obtained *in place* during the QR process (i.e., $\mathbf{Q}^{SR}$ does not need to be formed explicitly).

E. Update: New SLAM Features Initialization

When new SLAM features become available (i.e., points whose tracks span the entire window), the SR-ISWF adds them in the state vector and updates the information factor accordingly. Recall that at this point, the system's state vector

has the following structure [see (1) and (12)]:

$$\mathbf{x}_k^R = \begin{bmatrix}\mathbf{x}_S^T & \mathbf{x}_F^T\end{bmatrix}^T \tag{40}$$

and after adding the new SLAM features it becomes

$$\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_S^T & \mathbf{x}_S^{N^T} & \mathbf{x}_F^T\end{bmatrix}^T \tag{41}$$

where the new N_{NS} SLAM feature states, $\mathbf{x}_S^N = \begin{bmatrix}{}^G\mathbf{p}_{f_1}^T & \cdots & {}^G\mathbf{p}_{f_{N_{NS}}}^T\end{bmatrix}^T$, are appended to the end of the existing SLAM states, $\mathbf{x}_S$.

As shown in (19), the cost term corresponding to the information from the new SLAM feature measurements, $\mathcal{C}_{\mathbb{Z}_S}$, is split into two parts (17): $\mathcal{C}_{\mathbb{Z}_S^1}$ contains all the information on the features, while $\mathcal{C}_{\mathbb{Z}_S^2}$ involves only the poses. Hence, we use $\mathcal{C}_{\mathbb{Z}_S^1}$ in this step to initialize new SLAM features, while $\mathcal{C}_{\mathbb{Z}_S^2}$ will be used in the next step (see Sect. III-F) to perform updates. Specifically, from (38) and (15), initializing new SLAM features corresponds to the cost function:

$$\begin{aligned}\mathcal{C}_k^{NS}(\widetilde{\mathbf{x}}_k) &= \mathcal{C}_k^{SR}(\widetilde{\mathbf{x}}_k^R) + \mathcal{C}_{\mathbb{Z}_S^1}(\widetilde{\mathbf{x}}_S^N, \widetilde{\mathbf{x}}_F)\\ &= \|\mathbf{R}_k^{SR}\widetilde{\mathbf{x}}_k^R - \mathbf{r}_k^{SR}\|^2\\ &\quad + \sum_{j=1}^{N_{NS}} \|\mathbf{R}_f^{j\,G}\widetilde{\mathbf{p}}_{f_j} + \mathbf{F}_{1_S}^j\widetilde{\mathbf{x}}_F - \widetilde{\mathbf{z}}_{1_S}^j\|_{\sigma^2\mathbf{I}}^2\end{aligned} \tag{42}$$

If we partition the current upper-triangular information factor, $\mathbf{R}_k^{SR}$, and the corresponding residual vector, $\mathbf{r}_k^{SR}$, according to the state $\mathbf{x}_k^R$ in (40) as:

$$\mathbf{R}_k^{SR} = \begin{bmatrix}\mathbf{R}_{SS} & \mathbf{R}_{SF}\\ \mathbf{0} & \mathbf{R}_{FF}\end{bmatrix}, \quad \mathbf{r}_k^{SR} = \begin{bmatrix}\mathbf{r}_S\\ \mathbf{r}_F\end{bmatrix} \tag{43}$$

then the cost function in (42), after initializing the new SLAM features, becomes

$$\mathcal{C}_k^{NS}(\widetilde{\mathbf{x}}_k) = \|\mathbf{R}_k^{NS}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{NS}\|^2 \tag{44}$$

$$\mathbf{R}_k^{NS} = \left[\begin{array}{c|c|c} \mathbf{R}_{SS} & 0 & \mathbf{R}_{SF}\\ \hline & \frac{1}{\sigma}\mathbf{R}_f^1 & \frac{1}{\sigma}\mathbf{F}_{1_S}^1\\ 0 & \ddots & \vdots\\ & \frac{1}{\sigma}\mathbf{R}_f^{N_{NS}} & \frac{1}{\sigma}\mathbf{F}_{1_S}^{N_{NS}}\\ \hline 0 & 0 & \mathbf{R}_{FF}\end{array}\right] \tag{45}$$

$$\mathbf{r}_k^{NS} = \begin{bmatrix}\mathbf{r}_S\\ \hline \frac{1}{\sigma}\widetilde{\mathbf{z}}_{1_S}^1\\ \vdots\\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_{1_S}^{N_{NS}}\\ \hline \mathbf{r}_F\end{bmatrix} \tag{46}$$

Note that $\mathbf{R}_k^{NS}$ in (45) is already upper-triangular, since both $\mathbf{R}_{SS}$ and $\mathbf{R}_{FF}$ are upper-triangular [see (43)], and $\mathbf{R}_f^j$, $j = 1, ..., N_{NS}$, are upper-triangular from the Givens rotation operations in (18).

F. Update: New SLAM & MSCKF Pose Constraints

The last update step incorporates the pose-constraint information from both new-SLAM and MSCKF feature measurements. Specifically, [see (19), (44), and (16)], this step corresponds to the following changes to the cost function:

$$\begin{aligned}\mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) &= \mathcal{C}_k^{NS}(\widetilde{\mathbf{x}}_k) + \mathcal{C}_{\mathbb{Z}_S^2}(\widetilde{\mathbf{x}}_F) + \mathcal{C}_{\mathbb{Z}_M^2}(\widetilde{\mathbf{x}}_F)\\ &= \|\mathbf{R}_k^{NS}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{NS}\|^2 + \sum_{j=1}^{N_{NS}} \|\mathbf{F}_{2_S}^j\widetilde{\mathbf{x}}_F - \widetilde{\mathbf{z}}_{2_S}^j\|_{\sigma^2\mathbf{I}}^2\\ &\quad + \sum_{i=1}^{N_M} \|\mathbf{F}_{2_M}^i\widetilde{\mathbf{x}}_F - \widetilde{\mathbf{z}}_{2_M}^i\|_{\sigma^2\mathbf{I}}^2\end{aligned} \tag{47}$$

Note that both $\mathcal{C}_{\mathbb{Z}_S^2}$ and $\mathcal{C}_{\mathbb{Z}_M^2}$ involve only the pose state $\widetilde{\mathbf{x}}_F$, which is, by design, at the end of the state vector $\widetilde{\mathbf{x}}_k$ [see (41)]. After performing thin-QR factorization on the following stacked Jacobians corresponding to the pose part of the current factor:

$$\begin{bmatrix}\mathbf{R}_{FF}\\ \vdots\\ \frac{1}{\sigma}\mathbf{F}_{2_S}^j\\ \vdots\\ \frac{1}{\sigma}\mathbf{F}_{2_M}^i\\ \vdots\end{bmatrix} = \mathbf{Q}_{FF}^{\oplus}\mathbf{R}_{FF}^{\oplus}, \tag{48}$$

the cost function, in (47), becomes [see (45) and (46)]:

$$\mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) = \|\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{\oplus}\|^2 \tag{49}$$

$$\mathbf{R}_k^{\oplus} = \left[\begin{array}{c|c|c} \mathbf{R}_{SS} & 0 & \mathbf{R}_{SF}\\ \hline & \frac{1}{\sigma}\mathbf{R}_f^1 & \frac{1}{\sigma}\mathbf{F}_{1_S}^1\\ 0 & \ddots & \vdots\\ & \frac{1}{\sigma}\mathbf{R}_f^{N_{NS}} & \frac{1}{\sigma}\mathbf{F}_{1_S}^{N_{NS}}\\ \hline 0 & 0 & \mathbf{R}_{FF}^{\oplus}\end{array}\right] \tag{50}$$

$$\mathbf{r}_k^{\oplus} = \begin{bmatrix}\mathbf{r}_S\\ \hline \frac{1}{\sigma}\widetilde{\mathbf{z}}_{1_S}^1\\ \vdots\\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_{1_S}^{N_{NS}}\\ \hline \mathbf{r}_F^{\oplus}\end{bmatrix}, \quad \mathbf{r}_F^{\oplus} = \mathbf{Q}_{FF}^{\oplus^T}\begin{bmatrix}\mathbf{r}_F\\ \vdots\\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_{2_S}^j\\ \vdots\\ \frac{1}{\sigma}\widetilde{\mathbf{z}}_{2_M}^i\\ \vdots\end{bmatrix} \tag{51}$$

where (51) is computed *in place* through the QR process, i.e., without explicitly forming $\mathbf{Q}_{FF}^{\oplus}$.

Note that this step is very efficient for two reasons:

1) We only need to stack and perform QR on the pose Jacobian part [see (48)], instead of working with the whole factor.
2) Due to the upper-triangular structure of $\mathbf{R}_{FF}$, and the block upper-triangular structure of the Jacobians $\mathbf{F}_{2_S}^j$ and $\mathbf{F}_{2_M}^i$ (see Sect. II-C), a row permutation can bring the stacked matrix in (48) into a block upper-triangular form (with a large portion of zeros at the bottom left corner), which allows very efficient QR.

At this point, the information factor has been updated, into

$\mathbf{R}_k^{\oplus}$, using all the information inside the sliding window.

G. State Update

The last step in the SR-ISWF algorithm is to update the state, by minimizing (49) with respect to the error state vector:

$$\min_{\widetilde{\mathbf{x}}_k} \mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) = \min_{\widetilde{\mathbf{x}}_k} \|\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{\oplus}\|^2 \tag{52}$$

Since $\mathbf{R}_k^{\oplus}$ is upper-triangular and invertible, this corresponds to solving the linear equation

$$\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k = \mathbf{r}_k^{\oplus} \tag{53}$$

which simply requires an efficient back-substitution. Moreover, even faster solutions are achieved by noticing that the top-left portion of $\mathbf{R}_k^{\oplus}$, corresponding to the SLAM features, is block-diagonal [see (50)].

Finally, after solving for $\widetilde{\mathbf{x}}_k$, the state update is given by

$$\hat{\mathbf{x}}_k^{\oplus} = \hat{\mathbf{x}}_k + \widetilde{\mathbf{x}}_k \tag{54}$$

where $\hat{\mathbf{x}}_k$ comprises estimates for the states considered in the previous time step, $\hat{\mathbf{x}}_{k-1}$, as well as the new pose state estimate (from propagation) and the new SLAM feature estimates (from triangulation).

At this point, both the information factor $\mathbf{R}_k^{\oplus}$ and the state estimate $\hat{\mathbf{x}}_k^{\oplus}$ have been updated, and will serve as the prior state estimate and prior information factor, respectively, for the next time step.[5]

IV. Experimental Results

The goal of our experiments is to validate the capability of the proposed SR-ISWF algorithm for real-time operation on commercial-grade, resource-constrained devices, such as cell phones. We compare our algorithm to what is considered to be the state-of-the-art VINS in terms of accuracy and speed on mobile devices, the MSC-KF[6] [20, 10, 8], as well as our efficient single-precision implementation of the short-term smoother (STS) of [4], which is a square-root inverse sliding window filter that does not take advantage of the problem structure. For fair comparisons, all three algorithms use the same feature selection scheme and estimate the time-synchronization, rolling shutter, and extrinsic parameters.

A Samsung S4 mobile phone is used as our testbed. The S4 is equipped with a 1.6 GHz quad-core Cortex-A15 ARM CPU, a MEMS-quality IMU running at 100 Hz, and a rolling-shutter camera providing images with resolution 640×480 at 30 Hz. The camera and the IMU have separate clocks, while the image time-stamps are inaccurate.

We implemented a single-threaded pipeline consisting of feature extraction and tracking to provide visual measurements to the filter. First, the pipeline extracts 200 Harris corners [9] from images acquired at a frequency of 15 Hz, while these features are tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [18] across images. Then, a 2-Point RANSAC [13] is used for initial outlier rejection. The visual measurements are assumed to be contaminated by zero-mean white Gaussian noise with $\sigma = 1.5$ pixels. After that, the information manager selects a maximum of 20 SLAM and 60 MSCKF features, respectively. These feature measurements are then fed to the filter, which maintains a sliding window of 10 cloned poses. These clones are selected at a frequency of approximately 5 Hz. To achieve fast operation, our whole pipeline is running in single-precision floating-point format and is optimized for the ARM NEON co-processor.

TABLE I
Comparison: Loop-Closure Error Percentages

	Trajectory Length (m)	MSC-KF (%)	STS (%)	SR-ISWF (%)
Dataset 1	285	0.65	0.48	0.44
Dataset 2	56	0.52	0.71	0.77
Dataset 3	97	0.5	0.52	0.55
Dataset 4	105	0.58	0.74	0.7
Dataset 5	198	0.42	0.35	0.27

TABLE II
Comparison: Timing Results per Filter Update (msec)

	MSC-KF	STS	SR-ISWF
Filter Update Mean / Std	50.7 / 6.8	52.1 / 7.0	23.5 / 4.3
Total Pipeline Mean	114.4	100.2	71.8

A. Localization Accuracy

In the experiments, several indoor datasets were collected using the S4 cell phone sensors. In all the datasets, the device was returned back to its starting position. This allows us to quantitatively evaluate the accuracy of the estimators using the loop closure error percentage, which is computed as the ratio of the distance between the estimated starting and ending points against the total distance travelled (see Table I). In addition, to visualize the estimation accuracy, we overlay the trajectories onto the blueprints of the floor plans as reference.[7] Figs. 1 - 3 depict the trajectory estimates by the SR-ISWF and the MSC-KF. As evident, the estimated trajectories from the SR-ISWF and the MSC-KF differ in certain parts, but in most places they overlay each other. That, as well as the results in Table I lead us to believe that all three algorithms achieve comparable levels of accuracy.[8]

B. Computational Efficiency

Table II compares the processing times in milliseconds (msec) of the three algorithms running on the S4 mobile phone.

[5]The SR-ISWF allows iterative updates (i.e., re-linearization of all measurements), at each time step, hence reducing the effect of linearization errors. This requires repeating the process of the factor and state updates, after re-linearizing the measurements around the current, new, state estimates. In this case, the SR-ISWF becomes a sliding-window smoother.

[6]Specifically the version demonstrated at RSS 2014 [8].

[7]This is done based on the prior knowledge of the starting point and the initial heading direction, which are projected on the blueprints.

[8]We are currently in the process of preparing off-line, high-accuracy maps of the testing area, which will allow us to better quantify the errors of the estimators throughout their trajectories.

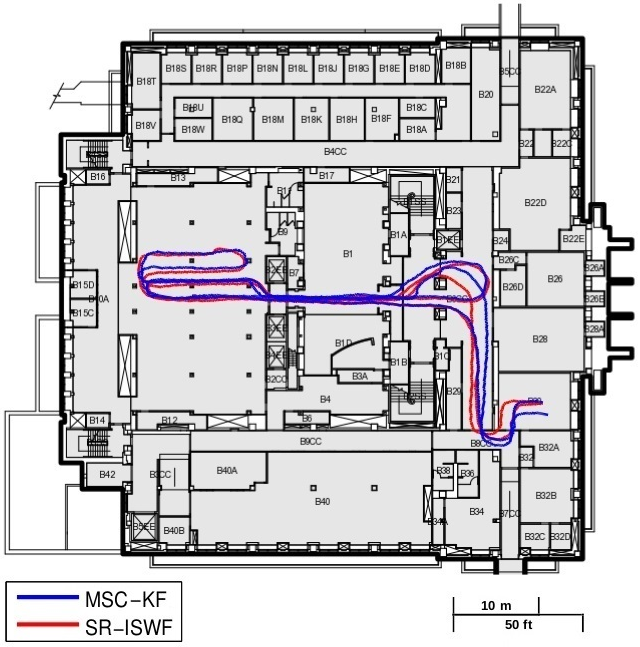

Fig. 1. Dataset 1: Overhead $x-y$ view of the estimated 3D trajectories.

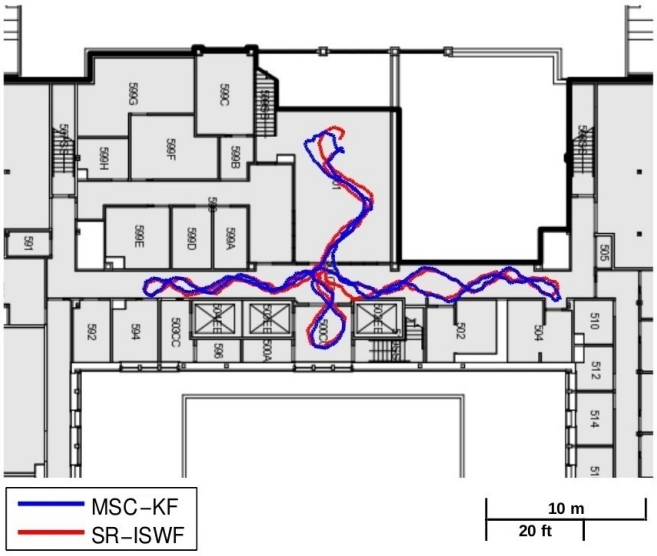

Fig. 2. Dataset 4: Overhead $x-y$ view of the estimated 3D trajectories.

As evident the SR-ISWF outperforms the other approaches in terms of speed (the filter update requires less than half of the time of either the MSC-KF or the STS), and achieves better than real-time operation.[9] This significant performance gain is attributed to the efficiency of the proposed SR-ISWF algorithm (see Sect. III), as well as the ability to operate in single-precision data format thanks to the square-root formulation.

Finally, we aim to compare our SR-ISWF with the iterative Kalman smoother (IKS) in [14]. The current optimized implementation of the IKS, however, can process MSCKF but not

[9]To the best of our knowledge, this is by far the fastest VINS algorithm on mobile devices, which if needed can run at full frame rate (e.g., fast motion).

Fig. 3. Dataset 5: Overhead $x-y$ view of the estimated 3D trajectories.

SLAM features. Thus, we are only able to provide qualitative comparison results: In terms of estimation accuracy, the two algorithms perform similarly under nominal conditions, yet the IKS is more robust to disturbances (e.g., abrupt changes in the time-synchronization due to changes in the lightening conditions, navigating through featureless areas, etc). In terms of speed, since the IKS employs iterative updates at each time-step, it runs slower than the MSC-KF (see Table II), thus, the IKS requires significantly more time than our SR-ISWF.

V. Conclusions

In this paper, we presented a square root inverse sliding window filter (SR-ISWF) for high-precision, real-time vision-aided inertial navigation systems (VINS) on resource-constrained mobile devices. Due to the square-root formulation, the proposed algorithm enjoys better numerical properties than an inverse filter (INVF), which enables using single-precision format for performing numerical operations very fast. We provided detailed derivations of all the steps of the SR-ISWF, which take advantage of the particular structure of the matrices involved, to deliver significant computational gain. A complete VINS with the proposed algorithm is implemented, using single precision format and ARM NEON instructions for specific steps. Experiments are carried out using a Samsung S4 mobile phone, and the proposed algorithm is compared with the state-of-the-art MSC-KF [8] and the short-term smoother in [4]. Our results show that, the SR-ISWF achieves comparable positioning accuracy with competing algorithms, while significantly outperforms them in terms of speed.

Acknowledgements

This work was supported by the University of Minnesota through the Digital Technology Center (DTC), AFOSR (FA9550-10-1-0567), and the National Science Foundation (IIS-1328722).

References

[1] Project Tango, https://www.google.com/atap/projecttango.

[2] Gerald J. Bierman. *Factorization Methods for Discrete Sequential Estimation*, volume 128 of *Mathematics in Science and Engineering*. Academic Press, New York, NY, 1977.

[3] Jean Y. Bouguet. Camera calibration toolbox for matlab. 2006. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[4] Han-Pang Chiu, Stephen Williams, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. Robust vision-aided navigation using sliding-window factor graphs. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 46–53, Karlsruhe, Germany, May 6 – 10 2013.

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

[6] Tue-Cuong Dong-Si and Anastasios I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 5655–5662, Shanghai, China, May 9 – 13 2011.

[7] Gene Golub and Charles Van Loan. *Matrix Computations*. JHU Press, 4th edition, 2013.

[8] Chao Guo, Dimitrios G. Kottas, Ryan DuToit, Ahmed Ahmed, Ruipeng Li, and Stergios I. Roumeliotis. Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In *Proc. of the Robotics: Science and Systems Conference*, Berkeley, CA, July 12 – 16 2014.

[9] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. of the Alvey Vision Conference*, pages 147–151, Manchester, UK, August 31 – September 2 1988.

[10] Joel A. Hesch, Dimitrios G. Kottas, Sean L. Bowman, and Stergios I. Roumeliotis. Towards consistent vision-aided inertial navigation. In *Proc. of the 10th International Workshop on the Algorithmic Foundations of Robotics*, pages 559–574, Cambridge, MA, June 13–15 2012.

[11] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, December 2008.

[12] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *International Journal of Robotics Research*, 31(2):216–235, February 2012.

[13] Laurent Kneip, Margarita Chli, and Roland Siegwart. Robust real-time visual odometry with a single camera and an imu. In *Proc. of the British Machine Vision Conference*, pages 16.1–16.11, Dundee, Scotland, August 29 - September 2 2011.

[14] Dimitrios G. Kottas and Stergios I. Roumeliotis. An iterative Kalman smoother for robust 3D localization on mobile and wearable devices. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 6336–6343, Seattle, Washington, May 26 – 30 2015.

[15] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334, December 2015.

[16] Mingyang Li and Anastasios I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation. In *Proc. of the Robotics: Science and Systems Conference*, pages 241–248, Sydney, Australia, July 9 –13 2012.

[17] Mingyang Li, Byung Hyung Kim, and Anastasios I. Mourikis. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4697–4704, Karlsruhe, Germany, May 6-10 2013.

[18] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the International Joint Conference on Artificaial Intelligence*, pages 674–679, Vancouver, British Columbia, August 24–28 1981.

[19] Peter S. Maybeck. *Stochastic models, estimation and control*, volume 1. Academic Press, New York, NY, 1979.

[20] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3482–3489, Rome, Italy, April 10–14 2007.

[21] Anastasios I. Mourikis and Stergios I. Roumeliotis. A dual-layer estimator architecture for long-term localization. In *Proc. of the Workshop on Visual Localization for Mobile Platforms*, pages 1 – 8, Anchorage, AK, June 24 - 26 2008.

[22] Esha D. Nerurkar, Kejian J. Wu, and Stergios I. Roumeliotis. C-KLAM: Constrained keyframe-based localization and mapping for long-term navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3638–3643, Hong Kong, China, May 31 – June 7 2014.

[23] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, August 2010.

[24] Nikolas Trawny and Stergios I. Roumeliotis. Indirect kalman filter for 3d attitude estimation. Technical report, University of Minnesota, Dept. of Comp. Sci. & Eng., March 2005.

[25] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proc. of the International Workshop on Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372, Corfu, Greece, September 21-22 1999.