Technical Report: Ultrafast Square-Root Filter-based VINS

Yuxiang Peng - yxpeng@udel.edu
Chuchu Chen - ccchu@udel.edu
Guoquan Huang - ghuang@udel.edu

Department of Mechanical Engineering
University of Delaware, Delaware, USA

Robot Perception and Navigation Group (RPNG)
Tech Report - RPNG-2024-SRF
Last Updated - March 5, 2024

Contents

Abstract

In this paper, we strongly advocate square-root covariance (instead of information) filtering for visual-inertial navigation, in particular on resource-constrained edge devices, because of its superior efficiency and numerical stability. Although Visual-Inertial Navigation Systems (VINS) have made tremendous progress in recent years, they still face resource stringency and numerical instability on embedded systems when imposing limited word length. To overcome these challenges, we develop an ultrafast and numerically-stable square-root filter (SRF)-based VINS algorithm (i.e., SR-VINS). The numerical stability of the proposed SR-VINS is inherited from the adoption of square-root covariance while the never-before-seen efficiency is largely enabled by the novel SRF update method that is based on our new permuted-QR (P-QR), which fully utilizes and properly maintains the upper triangular structure of the square-root covariance matrix. Furthermore, we choose a special ordering of the state variables which is amenable for (P-)QR operations in the SRF propagation and update and prevents unnecessary computation. The proposed SR-VINS is validated extensively through numerical studies, demonstrating that when the state-of-the-art (SOTA) filters have numerical difficulties, our SR-VINS has superior numerical stability, and remarkably, achieves efficient and robust performance on 32-bit single-precision float at a speed nearly twice as fast as the SOTA methods. We also conduct comprehensive real-world experiments to validate the efficiency, accuracy, and robustness of the proposed SR-VINS.

1 Introduction

Visual-Inertial Navigation Systems (VINS) that employ a single camera and an inertial measurement unit (IMU) to provide 3D motion tracking, have great potential in many applications such as AR/VR and robotics [1, 2, 3, 4]. VINS state estimation algorithms can be categorized into covariance and information forms. In the former such as the extended Kalman filter (EKF) and its variants, the estimator keeps tracking the dense covariance matrix to update the estimate [5, 6, 7, 8, 9, 10, 11]. In contrast, the information estimators such as extended Information filters (EIF) [12] or optimization-based methods [13, 14, 15, 16, 17, 18], maintain the information (Hessian) matrix and exploit its sparse structure in solving for estimates. However, both covariance and information filters face challenging numerical issues, in particular on resource-constrained edge platforms [19, 20], when limited word length (32-bit float, instead of 64-bit double) is available or it is required to achieve a potential speedup by leveraging SIMD (Single Instruction/Multiple Data) to vectorized matrix operations [21]. In the covariance form, the covariance matrix tends to lose its positive definiteness and cause the filter to diverge. In the information form, as the information matrix can easily become ill-conditioned (e.g., condition number larger than 10^9 [19]), naively inverting it during optimization would lead to large numerical errors (see Chapter 3.5.1 in [22]).

There exist methods that use the square root of the information matrix instead of its full matrix to mitigate the numerical instability and were shown to be effective to some extent in VINS [20, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. For example, the method in [20] maintains an upper triangular square root of prior information and uses QR-decomposition to incorporate new measurements into the prior, then invert it to solve for the state update. While this estimator achieves the same accuracy with the half of the word length, it still has the concerning numerical issue with a relatively high condition number ($> 10^5$) over time, especially when paired with a high-precision IMU [33, 34], resulting in substantial numerical inaccuracies that challenge long-term operations.

On the contrary, VINS estimators in the covariance form tend to offer better numerical stability. For instance, in the EKF-based VINS, the only matrix that typically requires inversion, the innovation covariance $\mathbf{S}$, usually possesses a good condition number [19]. By using the square-root covariance matrix, we can not only inherit the merits of the covariance form but also benefit from

square-root properties. Surprisingly, this idea of square-root filter (SRF) remains largely unexplored in VINS, primarily due to its inefficiency. Looking into history, the SRF has undergone significant improvements over the decades. Back in the 1960s, the initial SRF formulation was proposed by Potter and played a significant role in the Apollo project's success [35, 36], which has been extended to account for propagation (process) noise [37]. A key challenge is to improve its efficiency. Solutions include update methods using eigenvalue [38], Cholesky [39] and QR decomposition [40] within the SRF framework. However, compared with the conventional KF, these update methods in the SRF were shown to be less efficient, which is mainly because the triangular structure of the square-root covariance has been broken after the update. Agee [41] and Carlson [42] proposed update methods that maintain triangular structure and exhibit comparable efficiency to the KF. However, these methods are limited to sequential updates. In modern computers, batch updates involving vector operations are preferable, allowing for more level-3 BLAS[43] operations.

To address the aforementioned issues and fully utilize the benefit of the square-root covariance, in this work, we develop a novel (P-)QR-based SRF for VINS, termed SR-VINS. In particular, we propose a new permuted-QR (P-QR) decomposition that fully utilizes the upper-triangular structure during matrix factorization, which is theoretically shown to improve efficiency during batch updates. Additionally, when integrating into the sliding-window filtering framework, the proposed SR-VINS chooses a special ordering of the state variables which is amenable for (P-)QR operations in the SRF propagation and update and prevents unnecessary computation. Specifically, our main contributions can be summarized as follows:

- We propose a novel permuted-QR (P-QR) decomposition that not only fully utilizes the upper-triangular structure during matrix factorization, but also helps maintain the upper-triangular structure of the square-root covariance. With that, we develop an efficient (P-)QR-based SRF update method, which is shown to be significantly faster than the existing methods if $m > \frac{2}{3}n$ (where m and n are measurements and states size).
- We are among the first to design the SRF-based VINS with online calibration within an efficient sliding-window filter framework, which achieves never-before-seen efficiency and remarkable numerical stability even running on 32-bit. Our implementation demonstrates notable efficiency gain as it is almost two times faster than the state-of-the-art filters.
- We perform extensive numerical studies to highlight potential numerical challenges in VINS and underscore the advantages of our proposed SR-VINS. Real-world experiments validate the notable efficiency boost of the proposed method while maintaining accuracy.

2 Efficient Sqaure-Root Filtering

In comparison to a canonical EKF (or its variants) tracking the dense covariance matrix $\mathbf{P}$, the SRF propagates and updates the corresponding *upper triangular* square-root matrix $\mathbf{U}$, i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{P}$, while its state estimates are computed in the same way as the EKF [44]. By doing so, in principle, it possesses some key features that are particularly compelling to visual-inertial estimation at the edge. For example, the SRF can represent a broader dynamic range and reduce numerical errors by using a reduced condition number (i.e., the square root of the condition number of $\mathbf{P}$), thus offering better numerical stability. Moreover, the SRF can be significantly more efficient in both computation and memory consumption because it can use lower precision without sacrificing accuracy. Additionally, it automatically ensures the symmetry and positive semi-definite of the corresponding covariance matrix. However, in practice, it is not easy to capitalize these benefits if

blindly implementing the SRF, because the much-needed matrix triangulation operations required in the filter are computationally expensive. This is one of the reasons that the SRF has not been widely adopted in VINS, despite the aforementioned theoretical advantages. In this work, we, for the first time, fully take advantage of the upper-triangular structure of the SRF in VINS, by leveraging a new Permuted-QR (P-QR) algorithm.

2.1 Permuted-QR Decomposition

In contrast to the standard QR decomposition, for example, based on Givens rotation [45] or Householder [46], the proposed P-QR yields a lower, instead of upper, triangular matrix based on the following lemma:

Lemma 1. *For a full-rank matrix $\mathbf{M}_{m\times n}$ $(m > n)$, there exists the following lower-triangular P-QR decomposition:*[1]

$$\mathbf{M}_{m\times n} = \begin{bmatrix} \mathbf{A}_{(m-n)\times n} \\ \mathbf{B}_{n\times n} \end{bmatrix} \overset{P\text{-}QR}{=} \mathbf{Q}_{1_{m\times m}} \begin{bmatrix} \mathbf{0}_{(m-n)\times n} \\ \mathbf{F}_{n\times n} \end{bmatrix} \tag{1}$$

where $\mathbf{Q}_1$ is orthonormal and $\mathbf{F}$ is lower triangular.

Proof. We employ an anti-diagonal permutation matrix, $\mathbf{\Pi} = \texttt{adiag}([1\cdots 1]_n)$, and a row permutation matrix $\mathbf{\Gamma}$, to transform $\mathbf{M}$ into $\mathbf{M}''$ as:

$$\mathbf{M} := \mathbf{\Gamma}\mathbf{\Gamma}^\top\mathbf{M}\mathbf{\Pi}\mathbf{\Pi}^\top = \mathbf{\Gamma}\mathbf{\Gamma}^\top\mathbf{M}'\mathbf{\Pi}^\top = \mathbf{\Gamma}\mathbf{M}''\mathbf{\Pi}^\top \tag{2}$$

where $\mathbf{\Gamma}^\top$ is used to permute the rows of $\mathbf{M}'$ to make $\mathbf{M}''$ as close to upper triangular as possible so that we can perform QR decomposition on it efficiently. The standard QR of $\mathbf{M}''$ yields the following orthonormal matrix $\mathbf{Q}_2$ and upper triangular matrix $\mathbf{C}$:

$$\mathbf{M}'' \overset{\text{QR}}{=} \mathbf{Q}_2 \begin{bmatrix} \mathbf{C} \\ \mathbf{0} \end{bmatrix} \tag{3}$$

Substitution of $\mathbf{M}''$ into (2) yields the following identities:

$$\mathbf{M} = \mathbf{\Gamma}\mathbf{Q}_2 \begin{bmatrix} \mathbf{C} \\ \mathbf{0} \end{bmatrix} \mathbf{\Pi}^\top = \underbrace{\mathbf{\Gamma}\mathbf{Q}_2\mathbf{\Pi}'}_{\mathbf{Q}_1} \mathbf{\Pi}'^\top \begin{bmatrix} \mathbf{C} \\ \mathbf{0} \end{bmatrix} \mathbf{\Pi}^\top \tag{4}$$

$$= \mathbf{Q}_1\mathbf{\Pi}'^\top \begin{bmatrix} \mathbf{C} \\ \mathbf{0} \end{bmatrix} \mathbf{\Pi}^\top = \mathbf{Q}_1 \begin{bmatrix} \mathbf{0} \\ \mathbf{F} \end{bmatrix} \tag{5}$$

where we have employed a new anti-diagonal permutation matrix $\mathbf{\Pi}' = \texttt{adiag}([1\cdots 1]_m)$ along with $\mathbf{\Pi}$ to permute the upper triangular $\mathbf{C}$ into the lower triangular $\mathbf{F}$. □

Figure 1 visualizes how the matrix structure evolves during the proposed P-QR decomposition. Note that the resulting lower triangular structure of $\mathbf{F}$ will enable significant computation savings in the SRF update.

[1] Although we here assume the matrix $\mathbf{M}$ is of full column rank, our P-QR is applicable to rank-deficient matrices as the standard QR does.

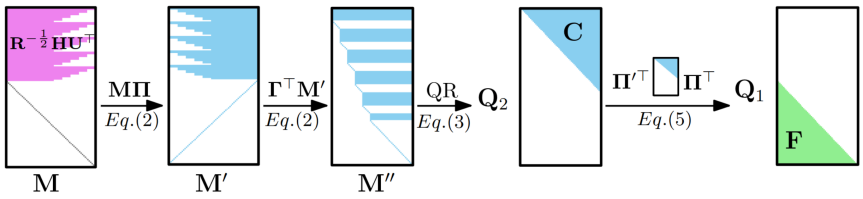

Figure 1: Evolution of the matrix structure when performing the proposed P-QR decomposition.

2.2 Propagation

2.2.1 Kalman Filter

The propagated (a prior) state and covariance estimate can be derived as follows:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{\Phi}_k \hat{\mathbf{x}}_{k|k} \tag{6}$$

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^{\top} + \mathbf{W}_k \tag{7}$$

2.2.2 Square Root Filter

The mean propagation of SRF remains the same as KF, here we introduce the special square root covariance propagation as:

$$\begin{bmatrix} \mathbf{W}_k^{\frac{1}{2}} \\ \mathbf{U}_{k|k}\mathbf{\Phi}_k^{\top} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}_k \begin{bmatrix} \mathbf{U}_{k+1|k} \\ \mathbf{0} \end{bmatrix} \tag{8}$$

where $\mathbf{W}_k = \mathbf{W}_k^{\frac{\top}{2}} \mathbf{W}_k^{\frac{1}{2}}$, and $\mathbf{W}_k^{\frac{1}{2}}$ can be obtained by Cholesky decomposition on $\mathbf{W}_k$.

It is clear from the above equation that we perform the QR decomposition of the LHS stacked matrix to get the propagated square root covariance matrix $\mathbf{U}_{k+1|k}$.

2.2.3 Proof

We first multiply LHS of Eq. (8) with its transpose:

$$\begin{bmatrix} \mathbf{W}_k^{\frac{\top}{2}} & \mathbf{\Phi}_k \mathbf{U}_{k|k}^{\top} \end{bmatrix} \begin{bmatrix} \mathbf{W}_k^{\frac{1}{2}} \\ \mathbf{U}_{k|k}\mathbf{\Phi}_k^{\top} \end{bmatrix} = \mathbf{\Phi}_k \mathbf{U}_{k|k}^{\top} \mathbf{U}_{k|k} \mathbf{\Phi}_k^{\top} + \mathbf{W}_k^{\frac{\top}{2}} \mathbf{W}_k^{\frac{1}{2}} \tag{9}$$

$$= \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^{\top} + \mathbf{W}_k \tag{10}$$

$$= \mathbf{P}_{k+1|k} \tag{11}$$

Then we multiply RHS of Eq. (8) with its transpose:

$$\begin{bmatrix} \mathbf{U}_{k+1|k}^{\top} & \mathbf{0} \end{bmatrix} \mathbf{Q}_k^{\top} \mathbf{Q}_k \begin{bmatrix} \mathbf{U}_{k+1|k} \\ \mathbf{0} \end{bmatrix} = \mathbf{U}_{k+1|k}^{\top} \mathbf{U}_{k+1|k} \tag{12}$$

$$= \mathbf{P}_{k+1|k} \tag{13}$$

From the equations above, we have shown the equivalence of KF and SRF propagation.

Remarks: Note that if most of the states have zero-dynamic, the QR decomposition performs in Eq. (8) only needs to work on serval states, which can be efficient.

2.3 State Augmentation and Clone

2.3.1 Kalman Filter

Given the prior state $\mathbf{x}_{k|k}$ and the process model. The state and covariance augmentation for the KF can be written as:

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k|k} \end{bmatrix} \tag{14}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top + \mathbf{W}_k & \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \\ \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top & \mathbf{P}_{k|k} \end{bmatrix} \tag{15}$$

We can then derive the cloning process, which is used in the MSCKF. We denote $\hat{\mathbf{x}}_{C,k+1|k}$ as the cloned state of $\hat{\mathbf{x}}_{k+1|k}$. Since they are identical, the state and the covariance can be derived as:

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{C,k+1|k} \\ \hat{\mathbf{x}}_{k|k} \end{bmatrix} \tag{16}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top + \mathbf{W}_k & \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top + \mathbf{W}_k & \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \\ \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top + \mathbf{W}_k & \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top + \mathbf{W}_k & \boldsymbol{\Phi}_k \mathbf{P}_{k|k} \\ \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top & \mathbf{P}_{k|k} \boldsymbol{\Phi}_k^\top & \mathbf{P}_{k|k} \end{bmatrix} \tag{17}$$

2.3.2 Square Root Filter

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k|k} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \mathbf{W}_k^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{U}_{k|k} \boldsymbol{\Phi}_k^\top & \mathbf{U}_{k|k} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q} \begin{bmatrix} \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1} \\ \mathbf{0} & \mathbf{U}_{k|k,2} \end{bmatrix} \tag{19}$$

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{C,k+1|k} \\ \hat{\mathbf{x}}_{k|k} \end{bmatrix} \tag{20}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{k|k,2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{21}$$

2.3.3 Proof

We first multiple both sides of (19) with theirs transpose

$$\begin{bmatrix} \mathbf{W}_k^{\frac{\top}{2}} & \mathbf{\Phi}_k \mathbf{U}_{k|k}^\top \\ \mathbf{0} & \mathbf{U}_{k|k}^\top \end{bmatrix} \begin{bmatrix} \mathbf{W}_k^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{U}_{k|k} \mathbf{\Phi}_k^\top & \mathbf{U}_{k|k} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{k+1|k}^\top & \mathbf{0} \\ \mathbf{U}_{k|k,1}^\top & \mathbf{U}_{k|k,2}^\top \end{bmatrix} \mathbf{Q}^\top \mathbf{Q} \begin{bmatrix} \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1} \\ \mathbf{0} & \mathbf{U}_{k|k,2} \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top + \mathbf{W}_k & \mathbf{\Phi}_k \mathbf{P}_{k|k} \\ \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top & \mathbf{P}_{k|k} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k|k,1} \\ \mathbf{U}_{k|k}^\top \mathbf{U}_{k+1|k,1} & \mathbf{U}_{k|k,1}^\top \mathbf{U}_{k|k,1} + \mathbf{U}_{k|k,2}^\top \mathbf{U}_{k|k,2} \end{bmatrix} \tag{23}$$

Then we can easily prove it through multiple (21) by its transpose on the left,

$$\mathbf{U}^\top \mathbf{U} = \begin{bmatrix} \mathbf{U}_{k+1|k}^\top & 0 & 0 \\ \mathbf{U}_{k+1|k}^\top & 0 & 0 \\ \mathbf{U}_{k|k,1}^\top & \mathbf{U}_{k|k,2}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{k|k,2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{24}$$

$$= \begin{bmatrix} \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k|k,1} \\ \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k+1|k}^\top \mathbf{U}_{k|k,1} \\ \mathbf{U}_{k|k,1}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1}^\top \mathbf{U}_{k+1|k} & \mathbf{U}_{k|k,1}^\top \mathbf{U}_{k|k,1} + \mathbf{U}_{k|k,2}^\top \mathbf{U}_{k|k,2} \end{bmatrix} \tag{25}$$

$$= \begin{bmatrix} \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top + \mathbf{W}_k & \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top + \mathbf{W}_k & \mathbf{\Phi}_k \mathbf{P}_{k|k} \\ \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top + \mathbf{W}_k & \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top + \mathbf{W}_k & \mathbf{\Phi}_k \mathbf{P}_{k|k} \\ \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top & \mathbf{P}_{k|k} \mathbf{\Phi}_k^\top & \mathbf{P}_{k|k} \end{bmatrix} \tag{26}$$

$$= \mathbf{P} \tag{27}$$

2.4 Update

Given the measurement equation $\mathbf{z}_{k+1} = \mathbf{H}_{k+1}\mathbf{x}_{k+1} + \mathbf{n}$, we now show the state mean and covariance update for both KF and SRF.

2.4.1 Kalman Filter

The state update equations for the Kalman filter are derived as:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\mathbf{r}_{k+1} \tag{28}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^\top \mathbf{S}_{k+1}^{-1} \mathbf{H}_{k+1}\mathbf{P}_{k+1|k} \tag{29}$$

where:

$$\mathbf{r}_{k+1} = \mathbf{z}_{k+1} - \mathbf{H}_{k+1}\hat{\mathbf{x}}_{k+1|k} \tag{30}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^\top + \mathbf{R}_{k+1} \tag{31}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^\top \mathbf{S}_{k+1}^{-1} \tag{32}$$

Kalman gain is first calculated and then used in both state and covariance updates.

2.4.2 Square Root Filter

During the update, a canonical SRF does not exploit the special structure of the square-root matrix update and incurs more expensive operations. In contrast, we propose a novel square-root update equation that is significantly more efficient by leveraging the proposed P-QR decomposition.

Table 1: FLOPs of the different measurement update assuming uncorrelated measurements and ignoring the terms of order lower than 3 (m measurements, n states).

Methods	**Potter**[35]	**Carlson**[42]	**Proposed**
Flops	$6mn^2$	$\frac{7}{2}mn^2$	$3mn^2 + \frac{1}{3}n^3$

Lemma 2. *It is equivalent to the KF update if the SRF updates its square-root covariance and state estimate as:*

$$\mathbf{U}_{k|k} = \mathbf{F}_k^{-\top}\mathbf{U}_{k|k-1} \tag{33}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{U}_{k|k}^{\top}\mathbf{U}_{k|k}\mathbf{H}_k^{\top}\mathbf{R}_k^{-1}\mathbf{r}_k \tag{34}$$

where $\mathbf{H}_k$ is the measurement Jacobian, $\mathbf{R}_k$ is the noise covariance, and $\mathbf{r}_k$ is the residual. Most importantly, $\mathbf{F}_k$ is lower triangular (and thus $\mathbf{F}_k^{-\top}$ is upper triangular), which is obtained by the following P-QR:

$$\begin{bmatrix} \mathbf{R}_k^{-\frac{1}{2}}\mathbf{H}_k\mathbf{U}_{k|k-1}^{\top} \\ \mathbf{I} \end{bmatrix} \overset{P\text{-}QR}{=} \mathbf{Q}\begin{bmatrix} \mathbf{0} \\ \mathbf{F}_k \end{bmatrix} \tag{35}$$

Note that the SRF first updates the square-root covariance $\mathbf{U}_{k|k}$ (33) and then use it to update the state $\hat{\mathbf{x}}_{k|k}$ (34).

2.4.3 Proof

We can now prove the equivalence between KF and SRF update equations:

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^{\top}\mathbf{S}_{k+1}^{-1}\mathbf{H}_{k+1}\mathbf{P}_{k+1|k} \tag{36}$$

$$= \mathbf{U}_{k+1|k}^{\top}\mathbf{U}_{k+1|k} - \mathbf{U}_{k+1|k}^{\top}\mathbf{U}_{k+1|k}\mathbf{H}_{k+1}^{\top}\left(\mathbf{H}_{k+1}\mathbf{U}_{k+1|k}^{\top}\mathbf{U}_{k+1|k}\mathbf{H}_{k+1}^{\top} + \mathbf{R}_{k+1}\right)^{-1}\mathbf{H}_{k+1}\mathbf{U}_{k+1|k}^{\top}\mathbf{U}_{k+1|k}$$

$$= \mathbf{U}_{k+1|k}^{\top}\left(\mathbf{I} - \mathbf{U}_{k+1|k}\mathbf{H}_{k+1}^{\top}\left(\mathbf{H}_{k+1}\mathbf{U}_{k+1|k}^{\top}\mathbf{U}_{k+1|k}\mathbf{H}_{k+1}^{\top} + \mathbf{R}_{k+1}\right)^{-1}\mathbf{H}_{k+1}\mathbf{U}_{k+1|k}^{\top}\right)\mathbf{U}_{k+1|k} \tag{37}$$

$$= \mathbf{U}_{k+1|k}^{\top}\left(\underbrace{\mathbf{I} + \mathbf{U}_{k+1|k}\mathbf{H}_{k+1}^{\top}\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1}\mathbf{U}_{k+1|k}^{\top}}_{\mathbf{F}_{k+1}^{\top}\mathbf{F}_{k+1}}\right)^{-1}\mathbf{U}_{k+1|k} \tag{38}$$

$$= \mathbf{U}_{k+1|k}^{\top}\mathbf{F}_{k+1}^{-1}\mathbf{F}_{k+1}^{-\top}\mathbf{U}_{k+1|k} \tag{39}$$

$$= \mathbf{U}_{k+1|k+1}^{\top}\mathbf{U}_{k+1|k+1} \tag{40}$$

From Eq. (37) to Eq. (38), the matrix inversion lemma is used.

Remarks: It is important to stress that the P-QR efficiently computes the lower-triangular matrix $\mathbf{F}_k$, which enables efficient update of the square-root covariance because both $\mathbf{F}_k^{-\top}$ and $\mathbf{U}_{k|k-1}$ are upper triangular. To see this, notice first that the LHS of (35) has an identity matrix at the bottom. Leveraging this structure allows for efficient QR decomposition because there is no need to zero out the elements below the diagonals of $\mathbf{I}$. When solving for $\mathbf{U}_{k|k}$, even though inverting $\mathbf{F}_k^{\top}$ is needed, thanks to its upper triangular structure, we can solve it efficiently using

back substitution on $\mathbf{F}_k^\top \mathbf{U}_{k|k} = \mathbf{U}_{k|k-1}$. With these structure benefits, we calculate the number of arithmetic operations required in the update with the assumptions: (i) measurements are uncorrelated, (ii) the terms that have orders smaller than 3 are ignored and (iii) the Householder algorithm is used to perform standard QR. Table 1 shows that our proposed SRF requires fewer operations than the most competitive Carlson update when $m > \frac{2}{3}n$ (m measurements, n states), which is often the case in VINS.

2.5 State Marginalization

In what follows, we will explain the state marginalization procedure in both KF and SRF with a simple example, where x is denoted as the state vector $\mathbf{P}$ represents the corresponding covariance and $\mathbf{U}$ is the upper-triangular square root covariance matrix. Note that we slightly abuse the notation to clarify the explanation.

2.5.1 Kalman Filter

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix} \tag{41}$$

- Marginalize $\mathbf{x}_3$:

$$\mathbf{x}_R = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \quad \mathbf{P}_R = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix} \tag{42}$$

- Marginalize $\mathbf{x}_2$:

$$\mathbf{x}_R = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{bmatrix}, \quad \mathbf{P}_R = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{13} \\ \mathbf{P}_{31} & \mathbf{P}_{33} \end{bmatrix} \tag{43}$$

2.5.2 Square Root Filter

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{U}_{22} & \mathbf{U}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{33} \end{bmatrix} \tag{44}$$

- Marginalize $\mathbf{x}_3$: In this case, we can simply grab the remaining square root covariance blocks as follows :

$$\mathbf{x}_R = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \quad \mathbf{U}_R = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22} \end{bmatrix} \tag{45}$$

- Marginalize $\mathbf{x}_2$: on the other hand, if we are going to marginalize the state in the middle, the marginalization process can be described as:

$$\mathbf{x}_R = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{U}_{23} \\ \mathbf{0} & \mathbf{U}_{33} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}'_{13} \\ \mathbf{0} & \mathbf{U}'_{23} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{46}$$

  where the remaining square root covariance matrix $\mathbf{U}_R$ is the upper part:

$$\mathbf{U}_R = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}'_{13} \\ \mathbf{0} & \mathbf{U}'_{23} \end{bmatrix} \tag{47}$$

2.5.3 Proof

Here we take the second case where $\mathbf{x}_2$ is marginalized as an example to prove.

Firstly, we show the equivalence between $\mathbf{P}$ and $\mathbf{U}$ before marginalization

$$\mathbf{U}^\top\mathbf{U} = \begin{bmatrix} \mathbf{U}_{11}^\top\mathbf{U}_{11} & \mathbf{U}_{11}^\top\mathbf{U}_{12} & \mathbf{U}_{11}^\top\mathbf{U}_{13} \\ \mathbf{U}_{12}^\top\mathbf{U}_{11} & \mathbf{U}_{12}^\top\mathbf{U}_{12} + \mathbf{U}_{22}^\top\mathbf{U}_{22} & \mathbf{U}_{12}^\top\mathbf{U}_{13} + \mathbf{U}_{22}^\top\mathbf{U}_{23} \\ \mathbf{U}_{13}^\top\mathbf{U}_{11} & \mathbf{U}_{13}^\top\mathbf{U}_{12} + \mathbf{U}_{23}^\top\mathbf{U}_{22} & \mathbf{U}_{13}^\top\mathbf{U}_{13} + \mathbf{U}_{23}^\top\mathbf{U}_{23} + \mathbf{U}_{33}^\top\mathbf{U}_{33} \end{bmatrix} \tag{48}$$

$$= \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix} \tag{49}$$

$$= \mathbf{P} \tag{50}$$

Then, we can prove the relationship between $\mathbf{P}_R$ and $\mathbf{U}_R$ after marginalization

$$\mathbf{U}_R^\top\mathbf{U}_R = \begin{bmatrix} \mathbf{U}_{11}^\top & \mathbf{0} \\ \mathbf{U}_{13}'^\top & \mathbf{U}_{23}'^\top \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{13}' \\ \mathbf{0} & \mathbf{U}_{23}' \end{bmatrix} \tag{51}$$

$$= \begin{bmatrix} \mathbf{U}_{11}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{U}_{13}'^\top & \mathbf{U}_{23}'^\top & \mathbf{0} \end{bmatrix} \mathbf{Q}^\top\mathbf{Q} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{13}' \\ \mathbf{0} & \mathbf{U}_{23}' \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{52}$$

$$= \begin{bmatrix} \mathbf{U}_{11}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{U}_{13}^\top & \mathbf{U}_{23}^\top & \mathbf{U}_{33}^\top \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{U}_{23} \\ \mathbf{0} & \mathbf{U}_{33} \end{bmatrix} \tag{53}$$

$$= \begin{bmatrix} \mathbf{U}_{11}^\top\mathbf{U}_{11} & \mathbf{U}_{11}^\top\mathbf{U}_{13} \\ \mathbf{U}_{13}^\top\mathbf{U}_{11} & \mathbf{U}_{13}^\top\mathbf{U}_{13} + \mathbf{U}_{23}^\top\mathbf{U}_{23} + \mathbf{U}_{33}^\top\mathbf{U}_{33} \end{bmatrix} \tag{54}$$

$$= \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{13} \\ \mathbf{P}_{31} & \mathbf{P}_{33} \end{bmatrix} \tag{55}$$

$$= \mathbf{P}_R \tag{56}$$

Remarks: As can be seen in eq. (45), when the marginalized state is at the end of the whole state, there is minimal effort to get the marginal covariance. If this is not the case as described in eq. (46), QR decomposition is required. But we should note that in the second case, only the second column block needs to perform QR since the first column block is already upper-triangular. Therefore in the design of SRF, placing the state that is likely to marginalize soon at the end would be recommended.

3 SRF-based VINS

3.1 State Vector

At time t_k, the system state $\mathbf{x}$ consists of the current navigation states $\mathbf{x}_{I_k}$, historical IMU pose clones $\mathbf{x}_C$, and a subset of 3D environmental (SLAM) features $\mathbf{x}_f$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{I_k}^\top & \mathbf{x}_{calib}^\top & \mathbf{x}_C^\top & \mathbf{x}_f^\top \end{bmatrix}^\top \tag{57}$$

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_k}^\top & {}^{G}\mathbf{v}_{I_k}^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top \end{bmatrix}^\top \tag{58}$$

$$\mathbf{x}_{calib} = \begin{bmatrix} t_d & {}^{I}_{C}\bar{q}^\top & {}^{C}\mathbf{p}_I^\top & \boldsymbol{\zeta}^\top \end{bmatrix} \tag{59}$$

$$\mathbf{x}_C = \begin{bmatrix} \mathbf{x}_{T_k}^\top \dots \mathbf{x}_{T_{k-c}}^\top \end{bmatrix}^\top \tag{60}$$

$$\mathbf{x}_f = \begin{bmatrix} \mathbf{f}_1^\top \dots \mathbf{f}_g^\top \end{bmatrix}^\top \tag{61}$$

where ${}^{I}_{G}\bar{q}$ is the unit quaternion (${}^{I}_{G}\mathbf{R}$ in rotation matrix form) that represents the rotation from the global $\{G\}$ to the IMU frame $\{I\}$; ${}^{G}\mathbf{p}_I$, ${}^{G}\mathbf{v}_I$, and ${}^{G}\mathbf{p}_{f_i}$ are the IMU position, velocity, and i'th feature position in $\{G\}$; $\mathbf{b}_g$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases; $\mathbf{x}_{T_i} = [{}^{I_i}_{G}\bar{q}^\top \ {}^{G}\mathbf{p}_{I_i}^\top]^\top$. t_d denote the time offset between camera and IMU, $\{{}^{I}_{C}\bar{q}^\top, {}^{C}\mathbf{p}_I^\top\}$ is the extrinsic between camera and IMU sensors and $\boldsymbol{\zeta}$ is the camera intrinsic parameters.

Remark: We have given careful consideration to the order of the state variables in the estimator [See Eq. (57)]:

- $\mathbf{x}_I$ and $\mathbf{x}_{calib}$ are prioritized at the top as they would not be marginalized.
- Clones $\mathbf{x}_C$ are ordered from the latest to oldest for easy marginalization of the oldest one.
- $\mathbf{x}_f$ is placed at the end for three reasons:
  1. SLAM features are marginalized frequently.
  2. This ordering makes the upper-triangular structure [$\mathbf{M}''$ in (3), see Figure 1] better preserved during P-QR when performing update.
  3. It ensures $\mathbf{U}$ is still upper-triangular after initializing a new SLAM feature

3.2 Propagation and Clone

The IMU kinematics are used to evolve the state from time t_k to t_{k+1}:

$${}^{I}_{G}\dot{\bar{q}}(t) = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}(t)){}^{I}_{G}\bar{q}(t) \tag{62}$$

$${}^{G}\dot{\mathbf{p}}_I(t) = {}^{G}\mathbf{v}_I(t) \tag{63}$$

$${}^{G}\dot{\mathbf{v}}_I(t) = {}^{I(t)}_{G}\mathbf{R}^\top\mathbf{a}(t) \tag{64}$$

$$\dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \tag{65}$$

$$\dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \tag{66}$$

where $\boldsymbol{\omega}(t) = [\omega_1\ \omega_2\ \omega_3]^\top$ and $\mathbf{a}(t)$ are the angular velocity and acceleration in the IMU local frame $\{I\}$; $\boldsymbol{\Omega}(\boldsymbol{\omega}(t)) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$ where $\lfloor\cdot\rfloor$ is the skew-symmetric matrix. $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ are white Gaussian noise that drive the IMU biases. A canonical three-axis IMU provides linear acceleration and angular velocity measurements, ${}^I\mathbf{a}_m$ and ${}^I\boldsymbol{\omega}_m$, expressed in the local IMU frame $\{I\}$ modeled as:

$$\mathbf{a}_m(t) = \mathbf{a}(t) - {}^I_G\mathbf{R}(t){}^G\mathbf{g} + \mathbf{b}_a(t) + \mathbf{n}_a(t) \tag{67}$$

$$\boldsymbol{\omega}_m(t) = \boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \tag{68}$$

where ${}^G\mathbf{g} \simeq [0, 0, -9.8]^\top$ is the gravitational acceleration expressed in $\{G\}$, $\mathbf{n}_g$ and $\mathbf{n}_a$ are zero-mean white Gaussian noise. ${}^I_G\mathbf{R}$ denotes the rotation matrix from global frame to local IMU frame. The IMU nonlinear kinematics can be formulated as follows:

$$\mathbf{x}_{I_{k+1}} = \mathbf{g}_I\left(\mathbf{x}_{I_k}, {}^I\mathbf{a}_k, {}^I\boldsymbol{\omega}_k, \mathbf{n}_I\right) \tag{69}$$

where $\mathbf{n}_I = [\mathbf{n}_g^\top\ \mathbf{n}_a^\top\ \mathbf{n}_{wg}^\top\ \mathbf{n}_{wa}^\top]^\top$. After linearization, the state translation matrix can be derived as [47]:

$$\boldsymbol{\Phi}_I(k+1,k) = \begin{bmatrix} \boldsymbol{\Phi}_{1,1} & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\Phi}_{1,4} & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{2,1} & \mathbf{I}_3 & \mathbf{I}_3\Delta t & \boldsymbol{\Phi}_{2,4} & \boldsymbol{\Phi}_{2,5} \\ \boldsymbol{\Phi}_{3,1} & \mathbf{0}_3 & \mathbf{I}_3 & \boldsymbol{\Phi}_{3,4} & \boldsymbol{\Phi}_{3,5} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{70}$$

with:

$$\boldsymbol{\Phi}_{1,1} = {}^{I_k}_{I_{k+1}}\hat{\mathbf{R}}^\top \qquad \boldsymbol{\Phi}_{1,4} = -\mathbf{J}_r\left({}^I\hat{\boldsymbol{\theta}}_{I_{k+1}}\right)\Delta t \qquad \boldsymbol{\Phi}_{2,1} = -\Delta\hat{\mathbf{R}}_k\lfloor\Xi_2\hat{\mathbf{a}}_k\rfloor \qquad \boldsymbol{\Phi}_{2,4} = \Delta\hat{\mathbf{R}}_k\Xi_4 \tag{71}$$

$$\boldsymbol{\Phi}_{2,5} = -\Delta\hat{\mathbf{R}}_k\Xi_2 \quad \boldsymbol{\Phi}_{3,1} = -\Delta\hat{\mathbf{R}}_k\lfloor\Xi_1\hat{\mathbf{a}}_k\rfloor \qquad \boldsymbol{\Phi}_{3,4} = \Delta\hat{\mathbf{R}}_k\Xi_3 \qquad \boldsymbol{\Phi}_{3,5} = -\Delta\hat{\mathbf{R}}_k\Xi_1 \tag{72}$$

and:

$$\boldsymbol{\Xi}_1 \triangleq \int_{t_k}^{t_{k+1}} \exp\left({}^I\hat{\boldsymbol{\omega}}\delta\tau\right)d\tau \qquad \boldsymbol{\Xi}_2 \triangleq \int_{t_k}^{t_{k+1}}\int_{t_k}^{s} \exp\left({}^I\hat{\boldsymbol{\omega}}\delta\tau\right)d\tau ds \tag{73}$$

$$\boldsymbol{\Xi}_3 \triangleq \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\mathbf{R}\lfloor{}^{I_\tau}\hat{\mathbf{a}}\rfloor\mathbf{J}_r\left({}^I\hat{\boldsymbol{\omega}}\delta\tau\right)\delta\tau d\tau \qquad \boldsymbol{\Xi}_4 \triangleq \int_{t_k}^{t_{k+1}}\int_{t_k}^{s} {}^{I_k}_{I_\tau}\mathbf{R}\lfloor{}^{I_\tau}\hat{\mathbf{a}}\rfloor\mathbf{J}_r\left({}^I\hat{\boldsymbol{\omega}}\delta\tau\right)\delta\tau d\tau ds \tag{74}$$

The noise Jacobian is derived as:

$$\mathbf{G}_I(t) = \begin{bmatrix} \boldsymbol{\Phi}_{1,4} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{2,4} & \boldsymbol{\Phi}_{2,5} & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{3,4} & \boldsymbol{\Phi}_{3,5} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3\Delta t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3\Delta t \end{bmatrix} \tag{75}$$

To perform IMU propagation in the SRF, we augment the inertial state by padding the new state at the top via stochastic cloning, i.e., $\begin{bmatrix}\mathbf{x}_{I_{k+1}}^T & \mathbf{x}_{I_k}^T\end{bmatrix}^T$, and propagate the corresponding square-root covariance via the standard QR [see (8)]:

$$\begin{bmatrix} \mathbf{W}_k^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{U}_k\boldsymbol{\Phi}_k^\top & \mathbf{U}_k \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}\begin{bmatrix} \mathbf{U}_{k+1} & \mathbf{U}_{k,1} \\ \mathbf{0} & \mathbf{U}_{k,2} \end{bmatrix} \tag{76}$$

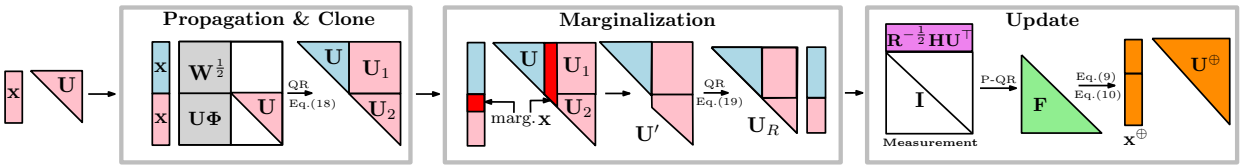

Figure 2: Visualization of the matrix structure during the SRF operations.

where $\mathbf{U}_k$ and $\mathbf{U}_{k+1}$ are the square-root covariance corresponding to the $\mathbf{x}_{I_k}$ and $\mathbf{x}_{I_{k+1}}$, respectively.

We are now to marginalize certain states such as the oldest state from the square-root covariance. As shown in Figure 2, we first remove the columns of $\begin{bmatrix}\mathbf{U}_{k+1} & \mathbf{U}_{k,1}\\ \mathbf{0} & \mathbf{U}_{k,2}\end{bmatrix}$ corresponding to the marginalized states and form $\mathbf{U}'$, and then perform QR of $\mathbf{U}'$ to obtain the upper-triangular square-root covariance $\mathbf{U}_R$:

$$\mathbf{U}' \overset{\text{QR}}{=} \mathbf{Q}_R \begin{bmatrix}\mathbf{U}_R \\ \mathbf{0}\end{bmatrix} \tag{77}$$

It is important to note that thanks to our special state ordering by placing non-marginalized variables at the top (e.g., $\mathbf{x}_I$ and $\mathbf{x}_{calib}$) and those to be marginalized (e.g., features) at the bottom, the resulting square-root covariance is close to the upper-triangular form, thus leading to significant computation savings.

3.3 Feature Update

The camera provides bearing observations of environmental 3D points. These observations can be used to update our state using the following measurement function (note that we use the anchored inverse depth feature model [48, 49]):

$$\mathbf{z} = \mathbf{h}({}^{C_k}\mathbf{p}_f(\mathbf{f}, \mathbf{x}_{T_k}, \mathbf{x}_{T_A}, {}^{I}_{C}\bar{q}, {}^{C}\mathbf{p}_I), \boldsymbol{\zeta}) + \mathbf{n}_k \tag{78}$$

$${}^{C_k}\mathbf{p}_f = {}^{C}_{I}\mathbf{R}\,{}^{I_k}_{G}\mathbf{R}\left({}^{I_A}_{G}\mathbf{R}^\top\,{}^{I_A}\mathbf{p}_f + {}^{G}\mathbf{p}_{I_A} - {}^{G}\mathbf{p}_{I_k}\right) + {}^{C}\mathbf{p}_I \tag{79}$$

$${}^{I_A}\mathbf{p}_f = \frac{1}{\rho}\begin{bmatrix}\cos(\theta)\sin(\phi)\\ \sin(\theta)\sin(\phi)\\ \cos(\phi)\end{bmatrix} \tag{80}$$

$$\mathbf{f} = \begin{bmatrix}\theta & \phi & \rho\end{bmatrix}^\top \tag{81}$$

where $\mathbf{h}({}^{C_k}\mathbf{p}_f, \boldsymbol{\zeta})$ is the pinhole camera model with radtan distortion [50], $\{\mathbf{I}_A\}$ is the anchor IMU frame, $\{\mathbf{I}_k\}$ is the IMU frame at the time k when this observation is captured by the camera.

The linearized measurement equation can be derived as:

$$\mathbf{r} = \mathbf{H}_I\tilde{\mathbf{x}}_I + \mathbf{H}_{calib}\tilde{\mathbf{x}}_{calib} + \mathbf{H}_f\tilde{\mathbf{x}}_f + \mathbf{n} =: \mathbf{H}_x\tilde{\mathbf{x}}_x + \mathbf{H}_f\tilde{\mathbf{x}}_f + \mathbf{n}_S \tag{82}$$

3.3.1 MSCKF Feature Measurement

If measurements are corresponding to the MSCKF feature $\mathbf{f}_m$ we project the linearized measurement function onto the left nullspace $\mathbf{N}$ of the feature Jacobian $\mathbf{H}_f$, to remove the feature dependency, as in the MSCKF [5]:

$$\mathbf{r}_m := \mathbf{N}^\top\mathbf{r} = \mathbf{N}^\top\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{N}^\top\mathbf{n} =: \mathbf{H}^m_x\tilde{\mathbf{x}}_x + \mathbf{n}_m \tag{83}$$

3.3.2 Delayed Initialization of SLAM Features

We now initialize a new SLAM feature $\mathbf{x}_{f_N}$ (instead of $\mathbf{x}_{f_O}$) given a set of measurements as in (82). We first perform P-QR to compress $\mathbf{H}_f^s$ to obtain the lower triangular matrix $\mathbf{H}'_{f_2}$:

$$\mathbf{H}_f^s \overset{\text{P-QR}}{=} \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix} \begin{bmatrix}\mathbf{0} \\ \mathbf{H}'_{f_2}\end{bmatrix} \tag{84}$$

Multiplying (82) by $[\mathbf{Q}_1\ \mathbf{Q}_2]^\top$ yields:

$$\begin{bmatrix}\mathbf{Q}_1^\top \\ \mathbf{Q}_2^\top\end{bmatrix} \mathbf{r}_s = \begin{bmatrix}\mathbf{0} & \mathbf{H}_x^i \\ \mathbf{H}'_{f2} & \mathbf{H}'_{x2}\end{bmatrix} \begin{bmatrix}\tilde{\mathbf{x}}_{f_N} \\ \tilde{\mathbf{x}}_x\end{bmatrix} + \begin{bmatrix}\mathbf{Q}_1^\top \\ \mathbf{Q}_2^\top\end{bmatrix} \mathbf{n}_s \tag{85}$$

$$\Rightarrow \begin{bmatrix}\mathbf{r}_i \\ \mathbf{r}'_2\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \mathbf{H}'_{f2}\end{bmatrix} \tilde{\mathbf{x}}_{f_N} + \begin{bmatrix}\mathbf{H}_x^i \\ \mathbf{H}'_{x2}\end{bmatrix} \tilde{\mathbf{x}}_x + \begin{bmatrix}\mathbf{n}_i \\ \mathbf{n}'_2\end{bmatrix} \tag{86}$$

We now can efficiently initialize the square-root covariance with the new feature being included in the state based on the bottom linear system $\mathbf{r}'_2$ of (86) as follows:

$$\mathbf{U}' = \begin{bmatrix}\mathbf{U} & \mathbf{U}\mathbf{H}_{x2}^{'-\top}\mathbf{H}_{f2}^{'-\top} \\ \mathbf{0} & \mathbf{R}^{'\frac{1}{2}}\mathbf{H}_{f2}^{'-\top}\end{bmatrix} \tag{87}$$

where $\mathbf{R}' = \mathbb{E}[\mathbf{n}'_2\mathbf{n}_2'^\top]$. The top linear system of (86) is used for SRF update as normal measurements.

3.3.3 SLAM Feature Reobservation

If measurements are corresponding to a SLAM feature $\mathbf{f}_O$ which has been initialized, the residual (82) is re-written as:

$$\mathbf{r}_s = \mathbf{H}_x^s\tilde{\mathbf{x}}_x + \mathbf{H}_{f_O}^s\tilde{\mathbf{x}}_{f_O} + \mathbf{n}_s \tag{88}$$

Remarks: Note that so far we only obtain all the measurement equations but have not used them to update the state and covariance yet. This is important because stacking all the measurements and processing them all at once is more efficient in the proposed SRF system.

3.3.4 Mahalanobis distance test

Mahalanobis distance test has to be used in practice in order to reject outliers, which is computed in the SRF: $d_m := \mathbf{r}^\top\left(\mathbf{H}\mathbf{U}^\top\mathbf{U}\mathbf{H}^\top + \mathbf{R}\right)^{-1}\mathbf{r}$. As the measurements of the MSCKF features $\mathbf{r}_m$ and SLAM feature initialization update $\mathbf{r}_i$ are not related to features (i.e., $\mathbf{H} = [\mathbf{H}_x\ \mathbf{H}_f] = [\mathbf{H}_x\ \mathbf{0}]$), we can compute:

$$\mathbf{U}\mathbf{H}^\top = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2 \\ \mathbf{0} & \mathbf{U}_3\end{bmatrix} \begin{bmatrix}\mathbf{H}_x^\top \\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_1\mathbf{H}_x^\top \\ \mathbf{0}\end{bmatrix} \tag{89}$$

Clearly, given the upper-triangular structure of $\mathbf{U}$ and the unique structure of the measurement Jacobian, we only need to compute $\mathbf{U}_1\mathbf{H}_x^\top$, instead of multiplying the measurement Jacobian $\mathbf{H}_x^\top$ with the full $\mathbf{U}$. For the SLAM feature update measurement $\mathbf{r}_s$, the sparsity of the measurement Jacobian which only relates to the corresponding IMU pose and feature allows us to leverage the upper-triangular structure of $\mathbf{U}$ to compute d_m more efficiently. Note that the computed $\mathbf{U}_1\mathbf{H}_x^\top$ can be used in the update to avoid redundant computation.

3.3.5 SRF Measurement Update

We perform batch update using all the MSCKF feature measurements $\mathbf{r}_m$ (83), SLAM feature initialization $\mathbf{r}_i$ (86), and SLAM feature measurements $\mathbf{r}_s$ (88). The stacked measurements are given by:

$$\begin{aligned}\begin{bmatrix}\mathbf{r}_m\\ \mathbf{r}_i\\ \mathbf{r}_s\end{bmatrix} &= \begin{bmatrix}\mathbf{H}_x^m\\ \mathbf{H}_x^i\\ \mathbf{H}_x^s\end{bmatrix}\tilde{\mathbf{x}}_x + \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{H}_{f_O}^s\end{bmatrix}\tilde{\mathbf{x}}_{f_O} + \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{x}}_{f_N} + \begin{bmatrix}\mathbf{n}_m\\ \mathbf{n}_i\\ \mathbf{n}_s\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{H}_x & \mathbf{H}_{f_O} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_x^\top & \tilde{\mathbf{x}}_{f_O}^\top & \tilde{\mathbf{x}}_{f_N}^\top\end{bmatrix}^\top + \mathbf{n}\end{aligned} \tag{90}$$

With this, we perform the SRF update as in (33) and (34). Note that the above measurement does not depend on the new SLAM feature and the corresponding Jacobian is zero, which can be leveraged to make the update even more efficient.

It is worth noting that this stacked measurement equation is not a function of new SLAM features $\mathbf{x}_{f_N}$, and the corresponding Jacobian is always zero. We can thus leverage this special structure of measurement Jacobian to make the filter update more efficient.

We first define the square-root covariance for the state before updating as:

$$\mathbf{U}^\ominus = \begin{bmatrix}\mathbf{U}_{xf}^\ominus & \mathbf{U}_{f_N,1}^\ominus\\ \mathbf{0} & \mathbf{U}_{f_N,2}^\ominus\end{bmatrix} \tag{91}$$

Following Eq. (35), $\mathbf{F}_{xf}$ can be derived as:

$$\begin{bmatrix}\mathbf{R}^{-\frac{1}{2}}\begin{bmatrix}\mathbf{H}_{xf} & \mathbf{0}\end{bmatrix}\mathbf{U}^{\ominus\top}\\ \mathbf{I}\end{bmatrix} = \begin{bmatrix}\mathbf{R}^{-\frac{1}{2}}\begin{bmatrix}\mathbf{H}_{xf} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{U}_{xf}^{\ominus\top} & \mathbf{0}\\ \mathbf{U}_{f_N,1}^{\ominus\top} & \mathbf{U}_{f_N,2}^{\ominus\top}\end{bmatrix}\\ \mathbf{I}\end{bmatrix} \tag{92}$$

$$= \begin{bmatrix}\mathbf{R}^{-\frac{1}{2}}\mathbf{H}_{xf}\mathbf{U}_{xf}^{\ominus\top} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{93}$$

$$\overset{\text{P-QR}}{=} \begin{bmatrix}\mathbf{Q}_{pqr1} & \mathbf{Q}_{pqr2} & \mathbf{0}\\ \mathbf{Q}_{pqr3} & \mathbf{Q}_{pqr4} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{F}_{xf} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{94}$$

$$= \mathbf{Q}\begin{bmatrix}\mathbf{0}\\ \mathbf{F}\end{bmatrix} \tag{95}$$

where

$$\mathbf{F} = \begin{bmatrix}\mathbf{F}_{xf} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix},\ \mathbf{Q} = \begin{bmatrix}\mathbf{Q}_{pqr1} & \mathbf{Q}_{pqr2} & \mathbf{0}\\ \mathbf{Q}_{pqr3} & \mathbf{Q}_{pqr4} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{96}$$

From Eq. (93) to Eq. (94), we apply P-QR on $\begin{bmatrix}\mathbf{R}^{-\frac{1}{2}}\mathbf{H}_{xf}\mathbf{U}_{xf}^{\ominus\top}\\ \mathbf{I}\end{bmatrix}$ such that

$$\begin{bmatrix}\mathbf{R}^{-\frac{1}{2}}\mathbf{H}_{xf}\mathbf{U}_{xf}^{\ominus\top}\\ \mathbf{I}\end{bmatrix} \overset{\text{P-QR}}{=} \begin{bmatrix}\mathbf{Q}_{pqr1} & \mathbf{Q}_{pqr2}\\ \mathbf{Q}_{pqr3} & \mathbf{Q}_{pqr4}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{F}_{xf}\end{bmatrix} \tag{97}$$

The updated square-root covariance matrix $\mathbf{U}^{\oplus}$ can be derived as as:

$$\mathbf{U}^{\oplus} = \mathbf{F}^{-\top}\mathbf{U}^{\ominus} \tag{98}$$

$$= \begin{bmatrix} \mathbf{F}_{xf}^{-\top} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{xf}^{\ominus} & \mathbf{U}_{f_N,1}^{\ominus} \\ \mathbf{0} & \mathbf{U}_{f_N,2}^{\ominus} \end{bmatrix} \tag{99}$$

$$= \begin{bmatrix} \mathbf{F}_{xf}^{-\top}\mathbf{U}_{xf}^{\ominus} & \mathbf{F}_{xf}^{-\top}\mathbf{U}_{f_N,1}^{\ominus} \\ \mathbf{0} & \mathbf{U}_{f_N,2}^{\ominus} \end{bmatrix} \tag{100}$$

where we note that the bottom right block $\mathbf{U}_{f_N,2}^{\ominus}$ remains the same before and after the update, and we only need to inverse $\mathbf{F}_{xf}$ instead of the whole $\mathbf{F}$ matrix.

Algorithm 1 `SR-VINS`

Propagation and Cloning: Propagate the IMU state while cloning the latest IMU pose [Eq. (76)] (*skip QR*)
Marginalization: Marginalize oldest clone and lost tracked SLAM features [Eq. (77)] (**QR**)
Measurements Formulation: Using the tracked features to formulate measurements and prepare for updates.

- MSCKF features via nullspace projection [Eq. (83)]
- SLAM feature initialization [Eq. (84),(86),(87)]
- SLAM features re-observation [Eq. (88)]

SRF update:

- Stack meas. [Eq. (90)] and do SRF update [Eq.(35)] (**P-QR**).

At this point, we have presented the main steps of the proposed SR-VINS as summarized in Algorithm 1.

3.4 Anchor Feature Anchor Change

Anchor feature representation has been commonly used in many state-of-the-art VINS algorithms [14, 49, 26]. The anchor frame, denoted as A, can be selected as any camera frame that observes the feature in the sliding window. When the anchor frame corresponding to a long-tracked slam feature is to be marginalized from the sliding window, it becomes necessary to perform an anchor change to represent the feature with respect to a new anchor frame in the state, enabling the consistent estimation of the feature. Here we present how to perform the anchor change in the SRF.

To facilitate the anchor change process, we seek to establish the relationship between the feature represented in the *old* anchor frame A_1, denoted as ${}^{A_1}\mathbf{p}_f$, and the new anchor frame A_2, denoted as ${}^{A_2}\mathbf{p}_f$. To simplify the explanation, we use the 3D anchor feature here as an example to perform anchor change, if other feature representations are used (e.g. 1D inverse depth representation [14], inverse depth representation [48], etc), extra Jacobians need to be calculated with respect to ${}^{A_1}\mathbf{p}_f$ and ${}^{A_2}\mathbf{p}_f$ and then applied with the chain rule. This relationship can be derived based on the fundamental principle that the global position of the feature, ${}^{G}\mathbf{p}_f$, remains unchanged regardless of the choice of anchor frame. Thus, we can establish the following equations to capture the relationship between the feature representations in the old and new anchor frames:

$${}^{G}\mathbf{p}_f = {}^{A_1}_{G}\mathbf{R}^{\top}{}^{A_1}\mathbf{p}_f + {}^{G}\mathbf{p}_{A_1} = {}^{A_2}_{G}\mathbf{R}^{\top}{}^{A_2}\mathbf{p}_f + {}^{G}\mathbf{p}_{A_2} \tag{101}$$

Linearization of the above equations, we can get:

$${}^{G}\tilde{\mathbf{p}}_f = \mathbf{H}_{old}\tilde{\mathbf{x}}_{old} + \mathbf{H}_{f_{old}}{}^{A_1}\tilde{\mathbf{p}}_f = \mathbf{H}_{new}\tilde{\mathbf{x}}_{new} + \mathbf{H}_{f_{new}}{}^{A_2}\tilde{\mathbf{p}}_f \tag{102}$$

where $\mathbf{x}_{old}$ and $\mathbf{x}_{new}$ denotes the orientation and position of the old anchor frame, $\{A_1\}$, and new anchor frame, $\{A_2\}$, respectively. $\mathbf{H}$ represents the Jacobians of the linearized system. Rearrange Eq. (102) we get:

$$ {}^{A_2}\tilde{\mathbf{p}}_f = \mathbf{H}_{f_{new}}^{-1}\mathbf{H}_{f_{old}}{}^{A_1}\tilde{\mathbf{p}}_f + \mathbf{H}_{f_{new}}^{-1}\mathbf{H}_{old}\tilde{\mathbf{x}}_{old} - \mathbf{H}_{f_{new}}^{-1}\mathbf{H}_{new}\tilde{\mathbf{x}}_{new} \tag{103} $$

We can then perform the SR-covariance propagation introduced in Section 2.2.2 to get the covariance for the new anchor feature ${}^{A_2}\mathbf{p}_f$. Note that this propagation is easier than the IMU propagation since no noise is involved.

3.5 Online calibration

In our SRF-based VINS, we also perform online spatiotemporal calibration of the camera-IMU time offset and extrinsic transformation, and camera intrinsic. One can simply take the derivative of the camera measurement function with respect to the desired variables that they wish to calibrate online (camera-IMU extrinsic and camera intrisic). In what follows, we focus on the time-offset calibration between the camera and IMU sensor.

Consider the IMU time t is corrupted by the time offset t_d. In analogy to [51], if a new image $\mathbf{z}_i(t)$ is received from the camera, we first employ IMU measurements to propagate up to time $t + \hat{t}_d$, where $\hat{t}_d$ is the estimate of time offset parameter, which can be found in [51]. After that, we augment and clone the state at time $t + t_d$, where we denoted as $\mathbf{x}(t + t_d)$. To simplify the explanation, we gave the following general linearized system:

$$ \tilde{\mathbf{x}}(t_k + t_d) = \tilde{\mathbf{x}}(t_k + \hat{t}_d) + \mathbf{H}_t\tilde{t}_d \tag{104} $$

where $\mathbf{H}_t$ is the Jacobian w.r.t $\tilde{t}_d$. Following the propagation method in SRF introduced in Section 2.2.2, we can derive the propagation Jacobian from Eq. (104) for the time offset calibration as:

$$ \mathbf{\Phi}_t = \begin{bmatrix} \mathbf{I} & & & \\ & \mathbf{I} & & \\ & \mathbf{H}_t & \mathbf{I} & \\ & & & \mathbf{I} \end{bmatrix} \tag{105} $$

4 Numerical Study

Table 2: Simulation parameters and prior standard deviations for measurement perturbations.

Parameter	Value	Parameter	Value
Gyro. White Noise	2.0e-4	Gyro. Rand. Walk	2.0e-5
Accel. White Noise	5.0e-4	Accel. Rand. Walk	4.0e-4
Cam Freq. (Hz)	10	IMU Freq. (Hz)	400
Num. Clones	11	Tracked Feat.	100
Max. MSCKF Feat.	40	Max. SLAM Feat.	50

We use a 30-minute, 2.4km UD-ARL trajectory (see Figure 3) and employ the OpenVINS simulator [49] to produce realistic visual bearings and inertial measurements, as detailed in Table 2. For a fair comparison across estimators, we build upon OpenVINS which utilizes EKF. We implemented float version of OpenVINS (EKF), the square-root inverse filter (SRIF), and the

Table 3: RMSE values for orientation (deg.) and position (m) based on 200 runs on UD-ARL with different estimators.

Methods	EKF	SRF	SRIF
double	0.957 / 0.146	0.957 / 0.146	0.957 / 0.146
float	0.960 / 0.146	0.959 / 0.146	1.045 / 0.174

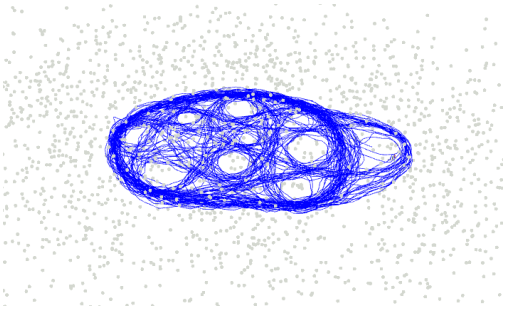

Figure 3: Simulated 2.4km UD-ARL trajectory.

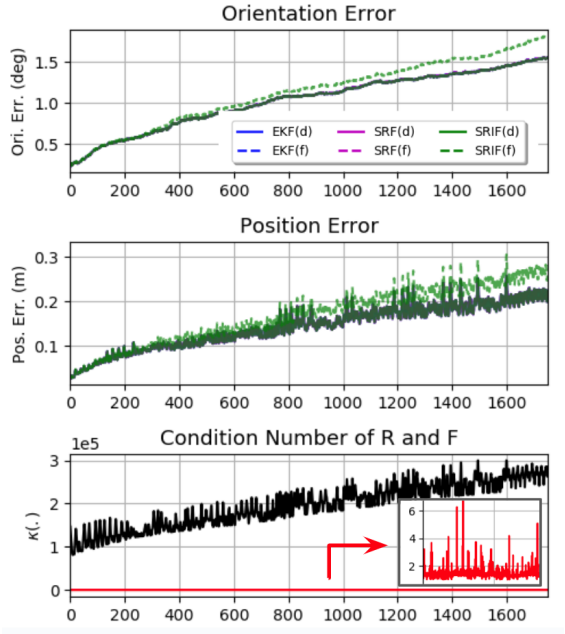

Figure 4: **Top:** Orientation/position errors of different estimators performed on UD-ARL dataset. 'd' is for double; 'f' is for float. While most estimators perform similarly and are hard to distinguish from the plot, SRIF(f) shows a clear drop in accuracy over time. **Bottom:** The condition number of the square-root information matrix (black line) with that of the P-QR lower triangular matrix **F** (red line, see (35)).

Table 4: Average Absolute Trajectory Error (ATE) in degrees/meters. 'd' and 'f' indicate the use of double and float. SRF(M) utilizes float and MSCKF features only for comparison with RVIO2.

Algo.	V101	V102	V103	V201	V202	V203	MH01	MH02	MH03	MH04	MH05
EKF(d)	0.70 / 0.06	1.67 / 0.06	2.88 / 0.07	0.95 / 0.10	1.38 / 0.06	1.28 / 0.14	1.74 / 0.10	0.91 / 0.17	1.14 / 0.12	0.95 / 0.25	1.03 / 0.41
EKF(f)	0.71 / 0.06	1.66 / 0.06	2.87 / 0.06	0.94 / 0.10	1.40 / 0.06	1.25 / 0.14	1.76 / 0.10	0.91 / 0.17	1.18 / 0.13	0.94 / 0.25	1.04 / 0.41
SRF(d)	0.68 / 0.05	1.68 / 0.06	2.88 / 0.06	0.99 / 0.11	1.40 / 0.06	1.28 / 0.14	1.75 / 0.10	0.93 / 0.18	1.18 / 0.12	0.95 / 0.24	1.04 / 0.39
SRF(f)	0.66 / 0.05	1.68 / 0.06	2.88 / 0.06	0.98 / 0.11	1.39 / 0.06	1.27 / 0.14	1.76 / 0.10	0.93 / 0.17	1.14 / 0.12	0.93 / 0.23	1.05 / 0.40
SRF(M)	0.63 / 0.08	1.75 / 0.06	1.76 / 0.08	0.74 / 0.10	1.36 / 0.08	1.19 / 0.16	1.56 / 0.15	0.95 / 0.22	1.02 / 0.17	1.12 / 0.25	0.93 / 0.39
RVIO2	0.88 / 0.09	2.27 / 0.10	2.02 / 0.10	2.19 / 0.13	1.90 / 0.11	1.50 / 0.15	2.60 / 0.17	1.00 / 0.15	1.08 / 0.19	1.10 / 0.24	0.95 / 0.32
VINS-Mono	0.82 / 0.07	2.74 / 0.10	5.15 / 0.15	2.13 / 0.09	2.57 / 0.13	3.43 / 0.29	0.78 / 0.20	0.86 / 0.18	1.82 / 0.23	2.51 / 0.41	0.94 / 0.29

proposed SRF-based VINS (SR-VINS). In Figure 4, the top two plots illustrate the orientation and position errors across different estimators with both double and float. Meanwhile, the bottom plot depicts the condition number of the square root information matrix and the condition number of the $\mathbf{F}$ matrix for SRF over time. Table 3 reports the average Root Mean Square Error (RMSE) for different estimators over 200 Monte-Carlo runs.

Given the covariance matrix $\mathbf{P}$, the square-root information matrix $\mathbf{R}$ is given by: $\mathbf{R}^\top\mathbf{R} = \mathbf{P}^{-1}$. From the figure, we observe as the condition number of $\mathbf{R}$ grows larger than $2e^5$, both orientation and position errors of SRIF(f) start showing a degraded performance compared to other filter design methods. This can also be seen in Table 3, the float SRIF is inaccurate with large RMSE values. This is likely due to the numerical issue when performing inversion on ill-conditioned $\mathbf{R}$ to solve for state update under limited machine precision (see Chapter 3.5.1 in [22]).

In contrast, the covariance-form estimators, both EKF and the proposed SRF, demonstrated consistent performance regardless of using double or float. This is evident from the comparable RMSE values in Table 3, as well as the consistent error trends in Figure 4. When performing the SRF update, the inversion of $\mathbf{F}$ is supposed to be the most numerically challenging operation. We thus plot its condition number shown in Figure 4 (bottom). Its condition number is shown to be stable and close to 1, demonstrating the improved numerical stability of the proposed SRF. Intuitively speaking, $\mathbf{F}^{-\top}$ is the transition matrix between $\mathbf{U}_{k|k-1}$ and $\mathbf{U}_{k|k}$. Therefore, as long as the measurements used in the update are not extremely accurate compared with the propagated estimation, we would expect $\mathbf{F}^{-\top}$ to be close to an identity matrix and be well-conditioned, which is almost always the case in the VINS in practice.

5 Real-World Experiments

We further evaluate the proposed SR-VINS on the EuRoC MAV dataset [52]. Only the left camera is used during the evaluation. The proposed system is built on top of OpenVINS[49]. We use the same default setup as OpenVINS [49], which extracts 200 sparse point features, keeps 11 clones, uses at most 50 SLAM features and 40 MSCKF features, performing camera-IMU extrinsic, time offset, and camera intrinsic calibration online. The proposed system is tested with both double and float versions, denoted as SRF(d) and SRF(f), respectively.[2]

We compare the proposed SR-VINS with the baseline Open-VINS (EKF(d)), which is originally in double, a float version of Open-VINS (EKF(f)) is also developed and evaluated. With default prior, float Open-VINS will experience negative diagonals in the covariance matrix and diverge in some sequences, thus its prior is tuned to make sure it runs on all the sequences. To make a fair comparison, all the versions of SRF and EKF use the same prior. We also compare with the open-sourced RVIO2 [27], which is a square-root inverse filter VIO based on robocentric state

[2] All computational results were performed in a single thread on an Intel(R) Core(TM) i7-11800H @ 2.30GHz.

Table 5: Average estimator run time (ms) comparison (excluding feature tracking) on EuRocMAV dataset. SRF(M) means adopting the same clone size and using only MSCKF features similar to the default setup of RVIO2 (15 clones, track 200 features, all the features are processed as MSCKF features once they lose track or reach maximum clone size).

Algorithm	EKF	SRF	SRF(M)	RVIO2	VINS-Mono
Double	4.2	2.8	1.7	-	22.4
Float	3.0	2.2	1.1	1.8	-

formulation, and VINS-Mono [14], which is an optimization-based sliding window VIO system. It is worth mentioning that EqVIO [8, 11] can also achieve impressive computational efficiency, however, its computation efficiency is gained by a much smaller state size, which is unable to fairly compare because its design principle is different from MSKCF-based VINS, thus, we do not include. The averaged Absolute Trajectory Error (ATE) values are reported in Table 4. Since RVIO2 only uses MSCKF features by default (i.e., no long-track SLAM features are maintained in the state vector, thus having a much smaller state size), we also report the float SRF performance with similar config (keeps 15 clones, tracks 200 features, all the features are processed as MSCKF features), denoted as SRF(M) in Table 4, for a fair comparison.

From Table 4, we can see that the performance of SRF(d), SRF(f), EKF(d), and EKF(f) are very similar as expected. The performance of double and float, EKF, and SRF are not exactly the same in the real world due to two reasons. First, χ^2 test is adopted to reject outliers and robustify the estimator and might introduce randomness. For example, in certain cases, SRF(d) might reject measurements that pass χ^2 test in SRF(f) because of slight numerical differences, this will cause SRF in different versions to use different measurements and have different performance. Second, OpenVINS (EKF) performs a "sequential" update, which first processes MSCKF features and then SLAM features for the consideration of efficiency, while SRF performs the update all at once. This also introduces differences in the state linearization points. Compared with RVIO2 and VINS-mono, SRF also achieves superior performance in almost all the sequences. Surprisingly, even SRF(M) achieves similar or even better performance than the other systems.

The efficiency of the estimators is also evaluated and reported in Table 5. Clearly, SRF(f) is much faster than its baseline EKF(d), reducing the runtime almost by half. Regardless of being in double or float format, SRF consistently prevails EKF. Remarkably, the double precision SRF even outperforms the float EKF. VINS-Mono runs the slowest as it performs iterative optimization. RVIO2 is also developed in float and shows excellent efficiency, but with a similar setup, SRF(M) in double prevails. Finally, SRF(M) achieves the best efficiency with 1.1 ms in estimator runs, which means it can run over 900Hz, especially suitable for running on a computation-constrained platform. The efficiency gain of SRF mainly comes from the proposed QR-based SRF update method, fully explored problem structure (state order, upper-triangular covariance, Jacobian structure, reusable computation).

6 Conclusions and Future Work

In this paper, we have developed the first square-root filter (SRF)-based VINS (i.e., SR-VINS) which significantly improves both the numerical stability and efficiency. We strongly advocate that the SRF is ideal for VINS due to its ability to represent a broader dynamic range, guarantee the property of the covariance matrix, reduce the memory requirement for covariance, and improve numerical stability. However, it is not trivial to capitalize on these advantages because of the challenge of its update inefficiencies, especially in dealing with large measurement sizes. To overcome

this issue and leverage the numerical advantage and the structure of the square-root covariance matrix, we have developed a novel permuted QR (P-QR)-based SRF update method. With this, we fully exploit the structure of the VINS problem to best utilize the upper triangular square-root covariance to gain never-before-seen speed boost. From our comprehensive numerical studies and real-world experiments, we have shown that the proposed SR-VINS can run robustly in float, gaining significant speedup (around 2 times faster than the SOTA filters), while exhibiting no accuracy loss, which makes it especially suitable for edging computing platforms. In the future, we are interested in further improving efficiency in visual tracking by leveraging the covariance matrix to reduce search space.

References

[1] Guoquan Huang. "Visual-Inertial Navigation: A Concise Review". In: *Proc. International Conference on Robotics and Automation*. Montreal, Canada, May 2019.

[2] Chuchu Chen, Yulin Yang, Patrick Geneva, Woosik Lee, and Guoquan Huang. "Visual-Inertial-aided Online MAV System Identification". In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Kyoto, Japan., 2022.

[3] Chuchu Chen, Patrick Geneva, Yuxiang Peng, Woosik Lee, and Guoquan Huang. "Monocular Visual-Inertial Odometry with Planar Regularitiee". In: *Proc. of the IEEE International Conference on Robotics and Automation*. London, UK., 2023.

[4] Yuxiang Peng, Chuchu Chen, and Guoquan Huang. "Quantized Visual-Inertial Odometry". In: *Proc. International Conference on Robotics and Automation*. Yokohama, Japan, May 2024.

[5] Anastasios I Mourikis and Stergios I Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation". In: *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE. 2007, pp. 3565–3572.

[6] Mingyang Li and Anastasios I Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711.

[7] Patrick Geneva, James Maley, and Guoquan Huang. "An Efficient Schmidt-EKF for 3D Visual-Inertial SLAM". In: *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, June 2019.

[8] Pieter van Goor and Robert Mahony. "An equivariant filter for visual inertial odometry". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 14432–14438.

[9] Yulin Yang, Chuchu Chen, Woosik Lee, and Guoquan Huang. "Decoupled right invariant error states for consistent visual-inertial navigation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1627–1634.

[10] Chuchu Chen, Yulin Yang, Patrick Geneva, and Guoquan Huang. "FEJ2: A Consistent Visual-Inertial State Estimator Design". In: *International Conference on Robotics and Automation (ICRA)*. Philadelphia, USA, 2022.

[11] Pieter van Goor and Robert Mahony. "EqVIO: An equivariant filter for visual-inertial odometry". In: *IEEE Transactions on Robotics* (2023).

[12] Guoquan Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. "An Observability Constrained Sliding Window Filter for SLAM". In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Francisco, CA, Sept. 2011, pp. 65–72. DOI: 10.1109/IROS.2011.6095161.

[13] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[14] Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[15] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. "Direct sparse visual-inertial odometry using dynamic marginalization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2510–2517.

[16] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[17] Chuchu Chen, Patrick Geneva, Yuciang Peng, Woosik Lee, and Guoquan Huang. "Optimization-based VINS: Consistency, Marginalization, and FEJ". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023.

[18] Chuchu Chen, Yuxiang Peng, and Guoquan Huang. "Fast and Consistent Covariance Recovery for Sliding-window Optimization-based VINS". In: *Proc. International Conference on Robotics and Automation*. Yokohama, Japan, May 2024.

[19] Dimitrios G Kottas and Stergios I Roumeliotis. "An iterative Kalman smoother for robust 3D localization on mobile and wearable devices". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 6336–6343.

[20] Kejian Wu, Ahmed M Ahmed, Georgios A Georgiou, and Stergios I Roumeliotis. "A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices." In: *Robotics: Science and Systems*. Vol. 2. Rome, Italy. 2015, p. 2.

[21] Michael J Flynn. "Very high-speed computing systems". In: *Proceedings of the IEEE* 54.12 (1966), pp. 1901–1909.

[22] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[23] Kejian J Wu and Stergios I Roumeliotis. "Inverse schmidt estimators". In: *Multiple Autonomous Robotic System Laboratory, Department of Computer Science & Engineering, University of Minnesota, Tech. Rep. Number-2016-003* (2016).

[24] David Caruso, Alexandre Eudes, Martial Sanfourche, David Vissiere, and Guy Le Besnerais. "An inverse square root filter for robust indoor/outdoor magneto-visual-inertial odometry". In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2017, pp. 1–8.

[25] Tong Ke, Kejian J Wu, and Stergios I Roumeliotis. "Rise-slam: A resource-aware inverse Schmidt estimator for slam". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 354–361.

[26] Zheng Huai and Guoquan Huang. "Markov Parallel Tracking and Mapping for Probabilistic SLAM". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Xi'an, China, 2021.

[27] Zheng Huai and Guoquan Huang. "Square-root robocentric visual-inertial odometry with online spatiotemporal calibration". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9961–9968.

[28] Frank Dellaert and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing". In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203.

[29] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: Fast incremental smoothing and mapping with efficient data association". In: *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE. 2007, pp. 1670–1677.

[30] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. "iSAM2: Incremental smoothing and mapping using the Bayes tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[31] Nikolaus Demmel, David Schubert, Christiane Sommer, Daniel Cremers, and Vladyslav Usenko. "Square root marginalization for sliding-window bundle adjustment". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13260–13268.

[32] Matthew W Givens and Jay W McMahon. "Square-Root Extended Information Filter for Visual-Inertial Odometry for Planetary Landing". In: *Journal of Guidance, Control, and Dynamics* 46.2 (2023), pp. 231–245.

[33] Paul Chauchat, Axel Barrau, and Silvere Bonnabel. "Factor graph-based smoothing without matrix inversion for highly precise localization". In: *IEEE Transactions on Control Systems Technology* 29.3 (2020), pp. 1219–1232.

[34] Paul Chauchat, Silvere Bonnabel, and Axel Barrau. "Invariant Smoothing with low process noise". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 4758–4763.

[35] James Potter and Robert Stern. "Statistical filtering of space navigation measurements". In: *Guidance and Control Conference*. 1963, p. 333.

[36] Richard H. Battin. "Astronautical Guidance". In: 1964.

[37] Paul Dyer and Stephen McReynolds. "Extension of square-root filtering to include process noise". In: *Journal of Optimization Theory and Applications* 3 (1969), pp. 444–458.

[38] JF Bellantoni and KW Dodge. "A square root formulation of the Kalman-Schmidt filter." In: *AIAA journal* 5.7 (1967), pp. 1309–1314.

[39] Angus Andrews. "A square root formulation of the Kalman covariance equations." In: *Aiaa Journal* 6.6 (1968), pp. 1165–1166.

[40] Paul Kaminski, A Bryson, and Stanley Schmidt. "Discrete square root filtering: A survey of current techniques". In: *IEEE Transactions on automatic control* 16.6 (1971), pp. 727–736.

[41] William S Agee and Robert H Turner. "Triangular decomposition of a positive definite matrix plus a symmetric dyad with application to kalman filtering". In: *White Sands Missile Range Tech. Rep* 38 (1972).

[42] Neal A Carlson. "Fast triangular formulation of the square root filter." In: *AIAA journal* 11.9 (1973), pp. 1259–1265.

[43] Jack J Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain S Duff. "A set of level 3 basic linear algebra subprograms". In: *ACM Transactions on Mathematical Software (TOMS)* 16.1 (1990), pp. 1–17.

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

[45] Wallace Givens. *Numerical computation of the characteristic values of a real symmetric matrix*. Tech. rep. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 1954.

[46] Alston S Householder. *The theory of matrices in numerical analysis*. Courier Corporation, 2013.

[47] Yulin Yang, B. P. W. Babu, Chuchu Chen, Guoquan Huang, and Liu Ren. "Analytic Combined IMU Integrator for Visual-Inertial Navigation". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Paris, France, 2020.

[48] Javier Civera, Andrew J Davison, and JM Martinez Montiel. "Inverse depth parametrization for monocular SLAM". In: *IEEE transactions on robotics* 24.5 (2008), pp. 932–945.

[49] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. "OpenVINS: A Research Platform for Visual-Inertial Estimation". In: *Proc. of the IEEE International Conference on Robotics and Automation*. Paris, France, 2020. URL: https://github.com/rpng/open_vins.

[50] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[51] Mingyang Li and Anastasios I Mourikis. "Online temporal calibration for camera–IMU systems: Theory and algorithms". In: *The International Journal of Robotics Research* 33.7 (2014), pp. 947–964.

[52] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163.