Fast and Consistent Covariance Recovery for Sliding-window Optimization-based VINS

Chuchu Chen, Yuxiang Peng, and Guoquan Huang

***Abstract*— In this paper, we introduce a novel and efficient technique for consistent covariance recovery in nonlinear optimization-based Visual-Inertial Navigation Systems (VINS). Estimating uncertainty in real-time is crucial for evaluating system performance and enhancing downstream operations such as data association. However accessing the marginal covariance of the state variables of interest in optimization-based VINS presents a significant challenge – a computational bottleneck due to the need to invert the high-dimensional information (Hessian) matrix. In our recent work [1], the First-Estimates Jacobian (FEJ) methodology was used to properly fix state linearization points in the optimization-based VINS, which seems counter-intuitive but improves the estimation performance in both consistency and accuracy. Capitalizing on this unique aspect of the FEJ strategy, in this work we carefully design the covariance recovery algorithm to improve efficiency by avoiding redundant computation. Remarkably, our approach achieves a computational speed that is 4-10 times faster than the existing methods. Through comprehensive numerical evaluations across four state-of-the-art marginalization archetypes, we not only affirm the consistency of our covariance estimates but underscore its superior computational efficiency.**

I. Introduction and Related Work

Visual-Inertial Navigation System (VINS) [2] has sparked significant research in the last decades and present immense potential in different domains such as autonomous navigation [3]–[7], extended reality [8], and space exploration [9]. By integrating a high-frequency inertial measurement unit (IMU) with a data-rich camera, VINS estimators can effectively estimate the sensing platform's pose generally using either efficient filter-based methods, performing linearization once [10]–[19], or optimization-based methods, which offer more precision through iteratively linearizing and solving but require greater computational resources [20]–[24].

Many existing VINS approaches offer an efficient and accurate mean estimation of the state vector. However, real-time uncertainty quantification (UQ) is also crucial for numerous practical applications. This is typically described by a marginal covariance matrix, representing the uncertainties between a subset of relevant state variables. For example, knowing the covariance can greatly simplify matching current measurements with their corresponding past observations in data association problems [25], [26]. Moreover, information-theoretic measures, such as mutual information derived from covariance, enhance robotics tasks like map merging [27], path planning [28], [29], graph sparsification [30]–[33], and active sensing [34].

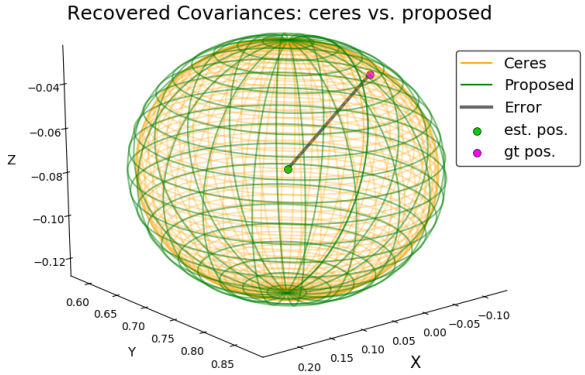

Fig. 1: An example of robot position covariance recovered by Ceres and proposed methods. Green and pink dots are the estimated and groundtruth positions, respectively. The black line shows the estimation error.

Another essential aspect in VINS is ensuring its estimation *consistency*[1]. First, from the system observability perspective, both filter-based [15], [16], [36], [37] and sliding-window optimization-based VINS [1], [38] have been observed to struggle with inconsistency issues arises from the *partially unobservable* and *nonlinear* nature of VINS. Discrepancies between sequential linearization points will cause information to gain along unobservable directions mistakenly and, in turn, significantly degrade the accuracy and reliability of the estimator. Second, overconfident covariance estimation underestimates the uncertainty associated with state estimates and could lead to incorrect data association, flawed decision-making, and potentially endangering the safety of autonomous robotic applications (i.e., obstacle avoidance). As such, it is crucial not just to guarantee a consistent estimate but also to recover the corresponding covariance that can properly describe the estimation error (see Fig. 1).

While computing covariance from filters, such as the Extended Kalman Filter (EKF), is relatively straightforward, the task becomes significantly more complex in optimization-based methods, which solve a Nonlinear Least-Squares (NLS) problem over a set of measurements iteratively. Typically, inverting the information matrix is necessary to recover the corresponding covariance, a process burdened by cubic complexity. Even if full covariance is not needed, accessing a subset of these variables in information form still requires marginalizing most state variables, adding to the

This work was partially supported by the University of Delaware (UD) College of Engineering, the Delaware NASA/EPSCoR Seed Grant, the NSF (MRI-2018905, SCH-2014264), Google ARCore, and Meta Reality Labs.

The authors are with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA. Email: {ccchu,yxpeng,ghuang}@udel.edu

[1]An estimator is consistent when its errors are zero-mean (unbiased) and the covariance matrix is equal to that reported by the estimator (see [35], Section 5.4).

complexity. Limited research initiatives aim to get covariance in optimization-based VINS or SLAM scenarios. Thrun et al. [39] suggest choosing a subblock of the information matrix of the desired state variables (i.e., the Markov blanket) and using its inverse (the conditional covariance) to approximate the actual covariance. The conditional covariance, however, is an over-confident approximation of the marginal covariance, can not consistently represent the true estimation uncertainty and could hurt downstream applications. An alternative method leverages a conservative approximation of the covariance matrix [40], [41], risks missing some actual information and decreasing constraints for ambiguous data association. Kaess and Dellaert [25] proposed to recover the exact covariance based on the incremental NLS solutions (i.e., iSAM [42]). Their method, relying on the recursive formula [43] utilized a dynamic programming algorithm to compute the marginal covariance from the square root information matrix [25]. SLAM++ further exploits the incremental nature of the solver and incrementally updates the marginal covariance as new measurements are integrated efficiently. However, it does not account for changes in the linearization points of the states [44], [45] . It is worth noting that these methods are specifically designed for incremental NLS solvers and cannot be directly applied to general optimization-based VINS.

Moreover, none of the aforementioned work have considered estimation consistency and state marginalization in covariance recovery. Our earlier work [1], [46] is, to the best of our knowledge, the first comprehensive study of VINS inconsistency within the context of a nonlinear factor graph. Using the First-estimates Jacobian (FEJ) design method, which may appear counter-intuitive, we properly fix certain state linearization points and significantly improve performance. In this paper, we build upon this foundation and introduce a novel, efficient, and consistent approach for covariance recovery in optimization-based VINS. In summary, our main contribution includes:

- We propose a novel, efficient, and consistent method for covariance recovery in sliding-window optimization-based VINS grounded on the FEJ method, demonstrating remarkable computational efficiency, around 4-10 times faster than existing methods.
- Based on the four most commonly used state-of-the-art marginalization schemes and their different FEJ design prerequisites, we ensure the consistency of both the state estimate and the recovered covariance.
- We validate the proposed method via extensive numerical studies, showing to achieve accurate covariance estimates and superior speed against existing approaches across various scenarios.

II. FEJ-based Optimization of VINS

We formulate the NLS problem over the entire trajectory until the current time t_k. The system state consists of the navigation states, $\mathbf{x}_{0:k}$, and 3D features, $\mathbf{x}_f$:

$$\mathbf{x}_{0:k} = \begin{bmatrix}\mathbf{x}_0^\top & \dots & \mathbf{x}_k^\top & \mathbf{x}_f^\top\end{bmatrix}^\top, \mathbf{x}_f = \begin{bmatrix}{}^G\mathbf{f}_1{}^\top & \dots & {}^G\mathbf{f}_g^\top\end{bmatrix}^\top$$
$$\mathbf{x}_k = \begin{bmatrix}{}^{I_k}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_g^\top & \mathbf{b}_a^\top\end{bmatrix}^\top \quad (1)$$

where ${}^I_G\bar{q}$ is the unit quaternion[2] that represents the rotation ${}^I_G\mathbf{R}$ from the global frame $\{G\}$ to the IMU frame $\{I\}$; ${}^G\mathbf{p}_I$ and ${}^G\mathbf{v}_I$ are the IMU position and velocity in $\{G\}$, respectively; $\mathbf{b}_g$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases; and the feature state $\mathbf{x}_f$ comprises the global position of g landmarks.

At timestep t_k, the batch maximum a posteriori (MAP) seeks to solve for the history of the state estimate $\hat{\mathbf{x}}_{0:k}$ by minimizing the cost function include: 1) prior $\mathcal{C}_{p_0}$ 2) IMU motion constraints $\mathcal{C}_I$, and 3) camera measurements $\mathcal{C}_f$:

$$\mathcal{C}(\mathbf{x}_{0:k}) = \mathcal{C}_{p_0} + \sum_{i=0}^{k-1}\mathcal{C}_{I_i} + \sum_{\mathbf{z}_{i,j}\in\mathcal{Z}_{0:k}}\mathcal{C}_{f_{ij}} \quad (2)$$

where the set $\mathcal{Z}_{0:k}$ denotes all measurements ($\mathbf{z}$) between $[t_0, t_k]$. To solve, we perform the Gauss-Newton iterative minimization. The second-order Taylor series of the l-th iteration with linearization point $\hat{\mathbf{x}}^l_{0:k}$ is:

$$\mathcal{C}(\hat{\mathbf{x}}^l_{0:k} \boxplus \delta\mathbf{x}^l_{0:k}) \simeq \mathcal{C}(\hat{\mathbf{x}}^l_{0:k}) + \mathbf{b}^l{}^\top\delta\mathbf{x}^l_{0:k} + \frac{1}{2}\delta\mathbf{x}^l{}^\top_{0:k}\mathbf{A}^l\delta\mathbf{x}^l_{0:k}$$

where $\mathbf{b}^l$ and $\mathbf{A}^l$ are the linearized gradient and Hessian. The correction term and the updated state $\hat{\mathbf{x}}^{l+1}_{0:k}$ can be solved by:

$$\mathbf{A}^l\delta\mathbf{x}^l_{0:k} = -\mathbf{b}^l \quad \Rightarrow \quad \hat{\mathbf{x}}^{l+1}_{0:k} = \hat{\mathbf{x}}^l_{0:k} \boxplus \delta\mathbf{x}^l_{0:k} \quad (3)$$

Given initial state $\hat{\mathbf{x}}_0$, this iterative algorithm will compute the global minimal estimates for the entire state $\mathbf{x}_{0:k}$ with all available measurements.

A. First-Estimates Jacobian (FEJ)

As robots explore the environment, solving the batch-MAP problem, Eq. (2) with all measurements becomes more computationally demanding with increasing state sizes. State marginalization is thus needed to manage complexity. However, this can introduce inconsistencies and lead to over-confident estimations, hurting accuracy and consistency. The FEJ technique was introduced to tackle this challenge [1], [46]. It evaluates the Hessian using the first estimate $\hat{\mathbf{x}}_R(k)$ instead of the current estimate $\hat{\mathbf{x}}_R(k')$ for all states $\mathbf{x}_R$ involved with the marginal:

$$\mathcal{C}(\mathbf{x}(k)) \simeq \mathcal{C}(\hat{\mathbf{x}}_R(k'), \hat{\mathbf{x}}_N(k')) + \mathbf{b}(\hat{\mathbf{x}}_R(k), \hat{\mathbf{x}}_N(k'))^\top\delta\mathbf{x}(k') + \frac{1}{2}\delta\mathbf{x}(k')^\top\mathbf{A}(\hat{\mathbf{x}}_R(k), \hat{\mathbf{x}}_N(k'))\delta\mathbf{x}(k') \quad (4)$$

where $\mathbf{x}_R$ denotes the remaining states that connected to the marginal state, $\mathbf{x}_N$ denotes the new state, and $\delta\mathbf{x}(k') = [\delta\mathbf{x}_R(k')^\top\ \delta\mathbf{x}_N(k')^\top]^\top$. For example, as shown in Figure 3 (top), after we marginalize the robot state $\mathbf{x}_0$ and feature $\mathbf{f}_2$, the remaining state $\mathbf{x}_1$, $\mathbf{f}_1$, $\mathbf{x}_3$, $\mathbf{x}_4$ and $\mathbf{x}_5$ connect to the marginalized prior factor $\mathbf{p}_1$ require to be FEJ (bottom left, shaded pink). For a thorough discussion on different marginalization methods and the application of FEJ please refer to our previous work [1], [46].

III. Covariance Recovery

Recovering the estimated covariance while solving the NLS [Eq. (2)] in real-time can be a computational bottleneck

[2]Throughout the paper, $\hat{\mathbf{x}}$ is used to denote the *current* best estimate of a random variable $\mathbf{x}$ with $\delta\mathbf{x} = \mathbf{x} \boxminus \hat{\mathbf{x}}$ denotes the error state. For the quaternion error state, we employ JPL multiplicative error [47] and use $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ defined by the error quaternion i.e., $\delta\bar{q} = \bar{q}\otimes\hat{\bar{q}}^{-1} \simeq [\frac{1}{2}\delta\boldsymbol{\theta}^\top\ 1]^\top$. The "$\boxplus$" and "$\boxminus$" operations map elements to and from a given manifold and equate to simple "+" and "-" for vector variables [48].

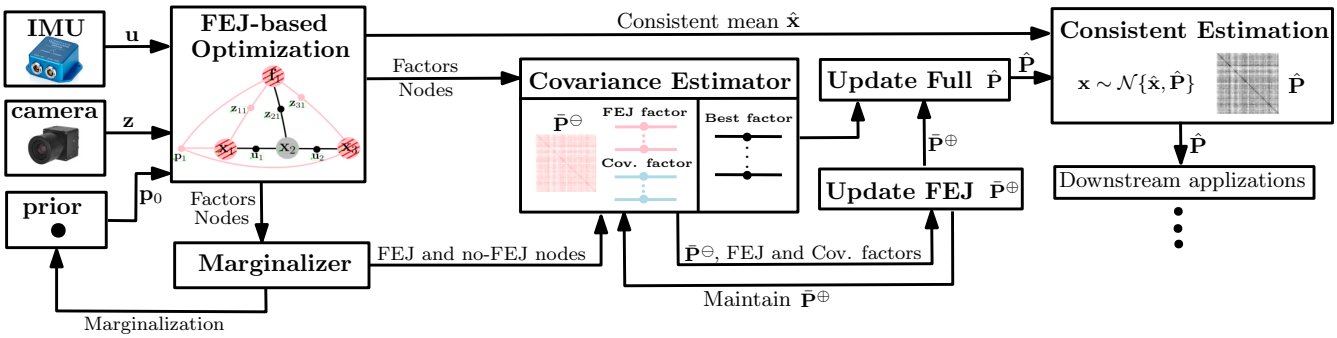

Fig. 2: Diagram of FEJ-based VINS within nonlinear optimization with efficient and consistent covariance recovery.

due to the need to invert the information matrix $\mathbf{A}$, such that $\mathbf{P} = \mathbf{A}^{-1}$. Naively performing matrix inversion will be cubic complexity, $O(n^3)$, where n is the dimension of states. In the following, we first explain the high-level idea of how the FEJ design method can accelerate this process by avoiding redundant computation in Section III-A. Next, we present how to recover covariance using IMU and camera factors in Section III-B. Section III-C explains in detail the proposed fast covariance recovery algorithm in conjunction with various marginalization methods.

A. FEJ-based Covariance Recovery

When solving the NLS problem with the FEJ method, some measurement functions and their corresponding Hessian matrices will be evaluated using fixed linearization points (i.e., first-state estimates). More importantly, when a measurement undergoes the FEJ process, even though the state estimates may continuously change due to new measurements or iterations, its first-estimate Jacobian and Hession information will remain *permanently fixed*. As shown in Figrue 3 (bottom left), the FEJ factors (colored in pink) and the corresponding covariances do *not* require re-evaluation.

Drawing from these insights, we first recover and maintain a "FEJ" covariance $\bar{\mathbf{P}}$, use the FEJ measurements:

$$\bar{\mathbf{P}}^{\oplus} = \Delta\mathbf{P}(\bar{\mathbf{P}}^{\ominus}, \bar{\mathbf{H}}, \bar{\mathbf{R}}) \tag{5}$$

where $\bar{\mathbf{P}}^{\ominus}$ and $\bar{\mathbf{P}}^{\oplus}$ denote the previous and updated FEJ covariance, $\Delta\mathbf{P}(\cdot)$ represents the update of FEJ covariance, which is a function of the first estimate Jacobian, $\bar{\mathbf{H}}$, and the measurement noise, $\bar{\mathbf{R}}$. We can then update the full covariance using the FEJ covariance and the rest of non-FEJ factors (i.e., best factors), its linearized Jacobian $\hat{\mathbf{H}}$ and corresponding noise matrix $\hat{\mathbf{R}}$ as:

$$\hat{\mathbf{P}}^{\oplus} = \Delta\mathbf{P}(\bar{\mathbf{P}}^{\oplus}, \hat{\mathbf{H}}, \hat{\mathbf{R}}) \tag{6}$$

where $\hat{\mathbf{P}}^{\ominus}$ and $\hat{\mathbf{P}}^{\oplus}$ are the full covariance before and after the update, respectively.

Algorithm 1 Efficient Covariance Recovery with FEJ

Build factor graph and perform optimization:
- Construct optimization problem using all measurements, linearize using the FEJ or best state estimates, and solve for the state mean estimates [Eq. (2), (3)].

State marginalization:
- Select states to be marginalized, set FEJ value for its connected remaining state and perform marginalization.

Recover FEJ covariance:
- Given the previous FEJ covariance $\bar{\mathbf{P}}^{\ominus}$ and linearized factors
  - **IMU factor:** propagate and augment [Eq. (8)].
  - **Camera factor:** select ones connect to FEJ features
    - ∗ **KEEP FEJ feature:** If is not in $\bar{\mathbf{P}}^{\ominus}$, initialize feature [Eq. (11),(12),(13)]; otherwise, EKF update.
    - ∗ **Other FEJ feature:** MSCKF update [Eq.(14)]
- Save and maintain the updated FEJ covariance, $\bar{\mathbf{P}}^{\oplus}$.

Recover full covariance:
- Given the updated FEJ covariance, $\bar{\mathbf{P}}^{\oplus}$, perform MSCKF updates with the rest of camera factors to get $\hat{\mathbf{P}}^{\oplus}$ [Eq.(14)].

B. Covariance Update Methods

We now explain how to derive $\boldsymbol{\Delta}\mathbf{P}(\cdot)$ with IMU and camera measurements. For simplicity, we do not distinguish FEJ and best covariance/Jacobians in the following explanations, as our methods are suitable for all.

1) IMU preintegration factor: Given the linearized IMU preintegration measurement:

$$\tilde{\mathbf{u}} \simeq -\boldsymbol{\Phi}_k\tilde{\mathbf{x}}_k + \tilde{\mathbf{x}}_{k+1} + \mathbf{w}_k \tag{7}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ denotes the propagated measurement noise. The augmented covariance can be derived as:

$$\mathbf{P}^{\oplus} = \begin{bmatrix} \boldsymbol{\Phi}_k\mathbf{P}^{\ominus}\boldsymbol{\Phi}_k^{\top} + \mathbf{Q}_k & \boldsymbol{\Phi}_k\mathbf{P}^{\ominus} \\ \mathbf{P}^{\ominus}\boldsymbol{\Phi}_k^{\top} & \mathbf{P}^{\ominus} \end{bmatrix} \tag{8}$$

where $\mathbf{P}^{\ominus}$ represent the covariance of $\mathbf{x}_k$.

Lemma 1: Given the same linearization point, the propagated and augmented covariance matrix, Eq. (8), is equivalent to inverting the information matrix.

Proof: See our tech report [49], Section 2.2.1. ■

2) Camera Measurement Factor: Given the robot states $\mathbf{x}$, their covariance $\mathbf{P}^{\ominus}$, feature $\mathbf{f}$, and its corresponding stacked measurements, $\mathbf{z}$, the linearized measurement equation is:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{H}_f\tilde{\mathbf{f}} + \mathbf{n} = \mathbf{H}\begin{bmatrix}\tilde{\mathbf{x}}^{\top} & \tilde{\mathbf{f}}^{\top}\end{bmatrix}^{\top} + \mathbf{n} \tag{9}$$

If the feature is already in the covariance, we can use Eq. (9) to perform an EKF covariance update:

$$\mathbf{P}^{\oplus} = \mathbf{P}^{\ominus} - \mathbf{P}^{\ominus}\mathbf{H}^{\top}\left(\mathbf{H}\mathbf{P}^{\ominus}\mathbf{H}^{\top} + \mathbf{R}\right)^{-1}\mathbf{H}_x\mathbf{P}^{\ominus} \tag{10}$$

where $\mathbf{R}$ is the measurement noise. If not, we can either a) initialize the feature, or b) do an MSCKF update, eliminating the need to maintain features and their state correlations to reduce the dimension of the covariance matrix.

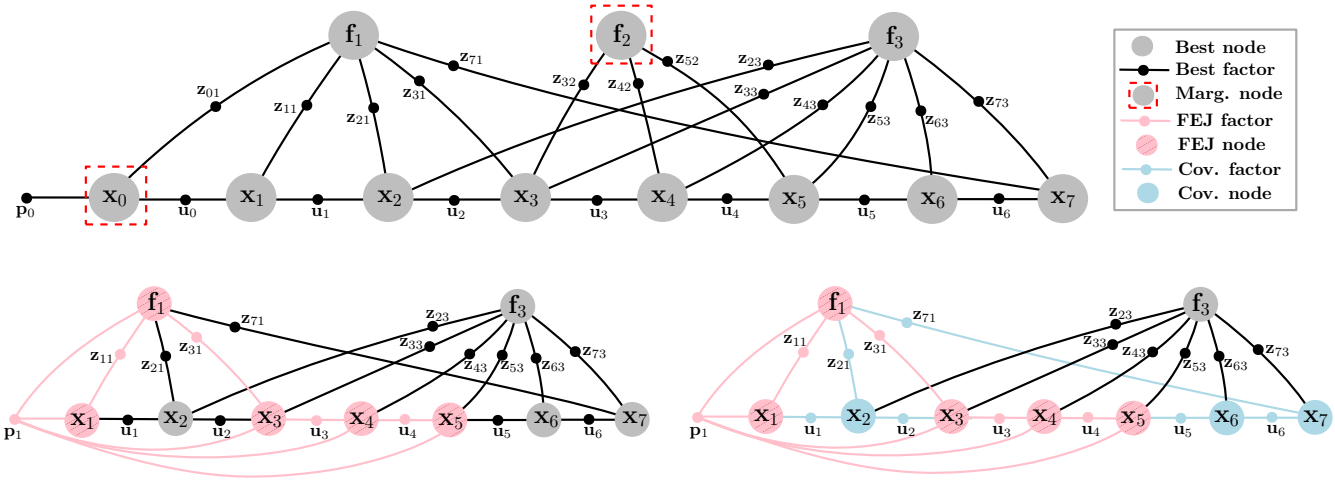

Fig. 3: **Top**: Example visual-inertial factor graph: Grey circles represent nodes (states) with edges (measurements) connecting related states. $\mathbf{f}_j$ is the jth feature, $\mathbf{x}_i$ is robot state at t_i. **Bottom left**: the resulting graph after marginalizing $\mathbf{x}_0$, $\mathbf{f}_2$, FEJ node with fixed linearization points are shaded in pink, while pink edges denote those evaluated and linearized with FEJ. Grey circles denote current state estimates (Best node), and black edges and measurements linearized with the current state estimates (Best factor). For details on performing FEJ, refer to [1]. **Bottom right**: node and factors used to recover the FEJ covariance. Blue are the extra "Cov." factor and nodes used besides the FEJ (pink) ones.

a) Feature Initialization: We first perform Givens rotations to $\mathbf{H}_f$ in Eq. (9) results in:

$$\begin{bmatrix}\tilde{\mathbf{z}}_{K1}\\ \tilde{\mathbf{z}}_{K2}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{x1}\\ \mathbf{H}_{x2}\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}\mathbf{H}_{f1}\\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{f}} + \begin{bmatrix}\mathbf{n}_{K1}\\ \mathbf{n}_{K2}\end{bmatrix} \tag{11}$$

The covariance is derived with the first sub-system, $\tilde{\mathbf{z}}_{K1}$:

$$\mathbf{P} = \begin{bmatrix}\mathbf{P}^{\ominus} & \mathbf{P}_{xf}\\ \mathbf{P}_{xf}^{\top} & \mathbf{P}_{ff}\end{bmatrix} = \begin{bmatrix}\mathbf{P}^{\ominus} & -\mathbf{P}^{\ominus}\mathbf{H}_{x1}^{\top}\mathbf{H}_{f1}^{-\top}\\ \mathbf{P}_{xf}^{\top} & \mathbf{P}_{ff}\end{bmatrix} \tag{12}$$

where $\mathbf{P}_{ff} = \mathbf{H}_{f1}^{-1}(\mathbf{H}_{x1}\mathbf{P}^{\ominus}\mathbf{H}_{x1}^{\top} + \mathbf{R}_1)\mathbf{H}_{f1}^{-\top}$. $\mathbf{R}_1$ is the measurement noise corresponding to $\mathbf{n}_{K1}$. We then perform the regular EKF update with the sub-system $\mathbf{z}_{K2}$ as:

$$\mathbf{P}^{\oplus} = \mathbf{P} - \mathbf{P}\mathbf{H}_{x2}^{\top}\left(\mathbf{H}\mathbf{P}\mathbf{H}_{x2}^{\top} + \mathbf{R}_2\right)^{-1}\mathbf{H}_{x2}\mathbf{P} \tag{13}$$

where $\mathbf{R}_2$ is the measurement noise corresponding to $\mathbf{n}_{K2}$.

b) MSCKF Feature: One can remove the feature by projecting the linearized measurement function, Eq. (9), onto the left nullspace $\mathbf{N}$ of $\mathbf{H}_f$:

$$\mathbf{N}^{\top}\tilde{\mathbf{z}} \simeq \mathbf{N}^{\top}\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{N}^{\top}\mathbf{n} \;\Rightarrow\; \tilde{\mathbf{z}}' = \mathbf{H}_x'\tilde{\mathbf{x}} + \mathbf{n}' \tag{14}$$

We can then directly perform EKF update:

$$\mathbf{P}^{\oplus} = \mathbf{P}^{\ominus} - \mathbf{P}^{\ominus}\mathbf{H}_x'^{\top}\left(\mathbf{H}_x'\mathbf{P}^{\ominus}\mathbf{H}_x'^{\top} + \mathbf{R}'\right)^{-1}\mathbf{H}_x'\mathbf{P}^{\ominus} \tag{15}$$

where $\mathbf{R}'$ is the measurement noise corresponding to $\mathbf{n}'$.

Lemma 2: Given the same linearized measurement equation, Eq. (9), the robot covariance will be the same if performing the above two methods in Section III-B.2.a and III-B.2.b.

Proof: See our tech report [49], Section 2.2.2.2. ■

Remark: While it is feasible to use the aforementioned "MSCKF-like" method for incremental covariance update. Naively recovering the covariance with every new measurement by incremental method without recognizing the redundant computation can still introduce significant overhead, as the EKF update complexity is $O(mn^2)$, where m is the dimension of measurements. Thanks to the FEJ method that fixes certain state linearization points, we can accelerate the covariance recovery process and eliminate the need to re-evaluate all the measurements redundantly.

C. Algorithm

Fig. 2 and Algorithm 1 summarize our proposed covariance recovery method. To properly implement FEJ, we fix the linearization point of states when they are about to be connected to the marginalized prior factor. When marginalizing certain robot states (i.e., the oldest/N-oldest) out of the sliding window, the four most commonly used methods are: 1) **KEEP**, marginalize the IMU, keep features; 2) **DROP**, drop factors between the feature and the to-be-marginalize IMU, keep the feature; 3) **MARG**, marginalize both IMU and its connected features; 4) **CKLAM**, duplicate and marginalize specific features and IMU. For a comprehensive understanding of FEJ implementation with these methods, refer to our earlier works [1], [46].

The optimal approach involves first recovering the FEJ covariance by leveraging fully FEJ'ed factors (all the states used to evaluate the factor's Jacobian are FEJ'ed). However, this introduces specific challenges and may even be unfeasible in certain scenarios. For example, shown in Fig. 3 (bottom left), using the fully FEJ'ed factors $\mathbf{z}_{01}$, $\mathbf{z}_{11}, \mathbf{z}_{31}$ to initialize feature $\mathbf{f}_1$ into covariance can be problematic due to the insufficient constraints like low parallax. Another example is when marginalize feature, $\mathbf{f}_2$, $\mathbf{z}_{32}$, $\mathbf{z}_{42}$ and $\mathbf{z}_{52}$ are used to create prior connect to $\mathbf{x}_3$, $\mathbf{x}_4$ and $\mathbf{x}_5$, which is not invertible, preventing us from obtaining the covariance.

Interestingly, our previous work found that applying FEJ to the IMU factors minimally affects estimation performance (see Table III in [1]). Thus, we select to use the IMU factor to update the FEJ covariance when it is initially added to the optimization problem. As depicted in Fig. 3 (bottom right), the blue IMU nodes and factors are "extra" just for recovering FEJ covariance. Notably, we still rigorously apply FEJ in optimization.

We also take particular care when integrating the camera factors. First, we can pinpoint the FEJ'ed features and their connected factors through marginalization. Different scenarios exist for FEJ features:

- **KEEP FEJ feature**: If a feature is FEJ'ed with the KEEP method (see Fig. 1(a) in [1]). We will initialize it, [See Section III-B.2.a] if it is not in the FEJ covariance; otherwise, an EKF update is performed to update the FEJ covariance;
- **Other FEJ features**: For all other FEJ features, we perform an MSCKF FEJ covariance update, Eq.(14).

It is worth noting that a minor discrepancy may arise between the linearization points used in the optimization and during feature covariance initialization. When initializing the feature, we employ all its linked factors to mitigate risks like low parallax, even if not all factors are fully FEJ'ed. For example, in Fig. 3 (bottom right), both pink (FEJ) and blue (extra) factors help to initialize $\mathbf{f}_1$. This discrepancy has limited impact as it only happens at initialization, and the covariance will be updated when the feature is reobserved. Furthermore, the KEEP-FEJ feature is the only one that requires initialization into the FEJ covariance because it can be reobserved and updated with new measurements. We will update and maintain the FEJ covariance when new IMU or FEJ camera factors are available.

We then use all the remaining no-FEJ camera factors to perform MSCKF updates and recover the full covariance, $\hat{\mathbf{P}}$, of the whole sliding window.

IV. Numerical Study

In line with our previous study [1], we simulate the realistic indoor dataset to test the proposed algorithm. We leverage the OpenVINS simulator [50], CPI preintegration [51] with the Ceres Solver [52] (see [49] for more detail about simulation parameters). In the following section, CKLAM shifts 2 robot states, others marginalize 1 each time.

A. Time Evaluation

We first investigate the computational cost, comparing the proposed covariance recovery method to the Ceres Solver. Ceres solver recovers covariance either using dense SVD or sparse QR decomposition of the information matrix followed by a back substitution, we focus on the sparse QR approach due to the impractical slowness of dense SVD. Shown in Fig. 6, we compared different covariance recovery algorithms (line colors), considering variances in sliding window sizes (x-axis) and the number of features incorporated into the optimization (line styles). "ceres" use Ceres to recover the latest IMU pose (6DoF). "ceres full" uses Ceres for full

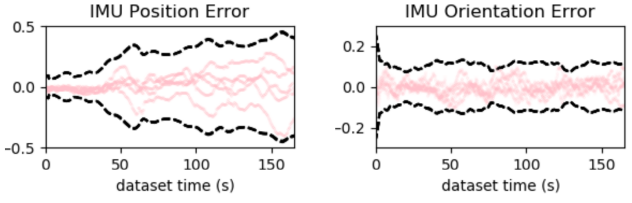

Fig. 4: IMU x position and roll angle errors (pink) with $\pm 3\sigma$ bounds (dashed black) across 20 Monte Carlo runs (KEEP).

TABLE I: Average ATE and NEES (20 runs)

Marg. method	ATE (deg/m)	NEES-ceres (ori/pos)	NEES-proposed (ori/pos)
KEEP	0.328 / 0.116	2.860 / 2.839	2.533 / 2.413
MARG	0.646 / 0.183	3.171 / 2.460	2.920 / 2.158
DROP	0.739 / 0.233	2.731 / 2.726	3.150 / 2.663
CKLAM	0.680 / 0.237	2.613 / 2.901	2.987 / 2.821

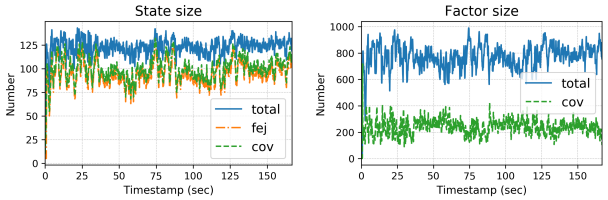

Fig. 5: The size of states (left) and factors (right). "total" represents the total size, "fej" is the number of FEJ nodes, and "cov" denotes the number of the states and factors we used to recover the covariance.

IMU states in the sliding window, while "proposed" is our proposed FEJ-based method. Given space constraints, we show the runtime for each algorithm only with a feature size of 100. For detailed results, refer to our technical report [49][3].

Generally, an increase in state or measurement size corresponds to a longer time. Recovering the full IMU states within the sliding window is more time-consuming than recovering the latest IMU pose via Ceres. Throughout our tests, our proposed method is the most efficient—evidenced by the green lines anchoring the bottom in all figures. Impressively, we outperform "ceres", despite recovering covariance for the full sliding-window states, whereas "ceres" is limited to the latest 6DoF pose, highlighting our approach's practical advantages. Among the marginalization techniques, the KEEP method is the slowest for "ceres" due to the denser prior factor. Interestingly, looking into the proposed method, it has the most significant efficient gain in the KEEP method because a substantial number of features are FEJ'ed during the marginalization.

Fig. 5 (left) shows total, FEJ state sizes, and states for covariance recovery (pink and blue nodes in Fig. 3) for KEEP method when the number of features is 100. As discussed in Section III-C, there is a small difference in the number of FEJ and Cov nodes. The right figure shows the total number of factors in the optimization problem and the factors used to update covariance at every timestep. It is clear to see that every time we only use a subset of total factors to update covariance, this is because FEJ'ed factors have been absorbed into FEJ covariance, which does not require re-evaluation. This shows the proposed method avoids redundant computation.

B. Consistency Evaluation

We then look into the consistency of the system and the recovered covariance. In this section, we present simulation

[3]All computational results were performed in a single thread on an 12th Gen Intel(R) Core(TM) i7-12800HX.

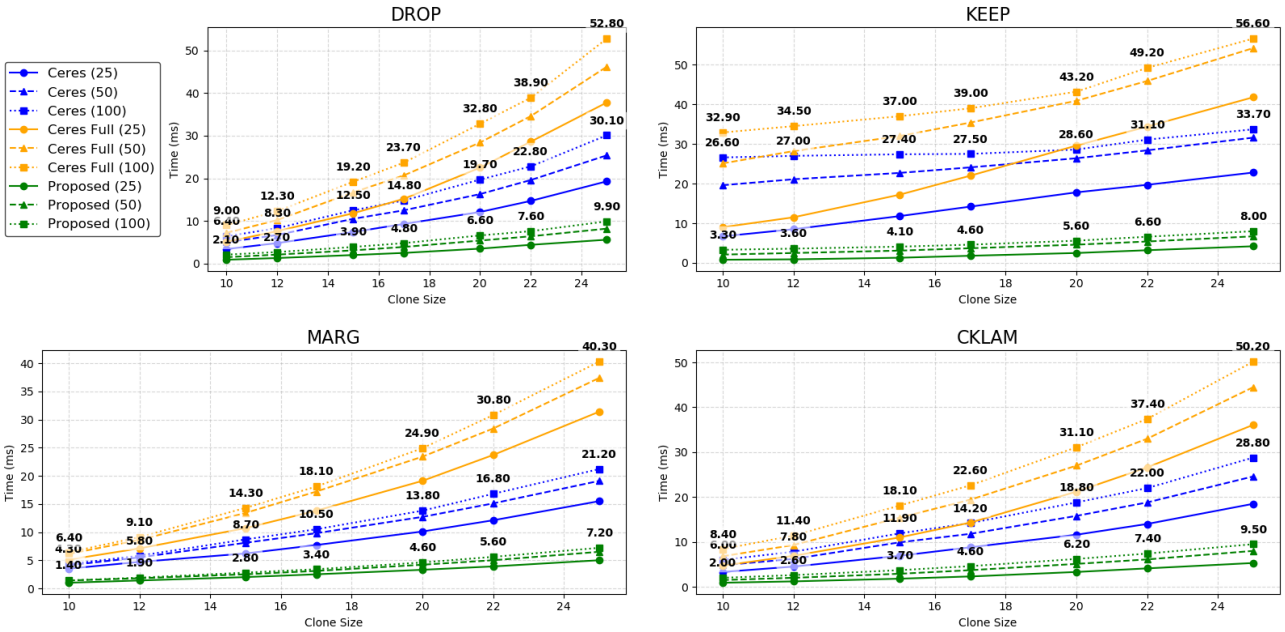

Fig. 6: Timing comparison in covariance recovery across algorithms with varied marginalization methods. Line colors denote the different algorithms. "**Ceres (blue)**": utilizes Ceres to recover covariance of the most recent IMU pose (6 DoF), "**Ceres full (orange)**": use Ceres to recover all IMU states within the sliding window, "**Proposed (green)**" proposed FEJ-based covariance recovery method. Different line style denotes the different number of feature in the optimization problem (25, 50, and 100 features). x-axis is the clone (sliding window) size, while y-axis is the computional time (ms).

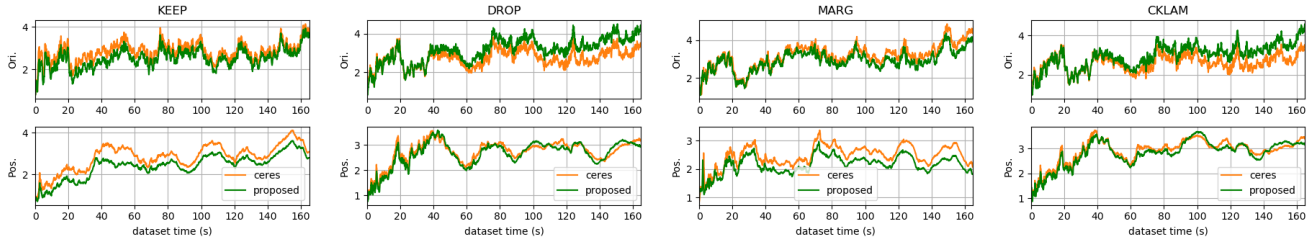

Fig. 7: NEES for IMU orientation & position over 20 runs: Ceres vs. proposed

results incorporating 35 features within the sliding window. The metrics used are Absolute Trajectory Error (ATE) and Normalized Estimation Error Squared (NEES), which should match the 3 DoF state size for both orientation and position if the estimator is consistent. Table I reports the ATE and NEES calculated from covariance recovered by ceres and the proposed method over 20 Monte Carlo runs.

All marginalization techniques exhibit consistency, with NEES values close to 3. The discrepancies between Ceres and our proposed method arise from the slightly varied linearization points used in each, as detailed in Section III-C. The KEEP method, due to its longer feature tracking, unsurprisingly achieves the most accurate performance with the smallest ATE. Fig. 7 presents the average NEES of the IMU pose across 20 Monte Carlo simulations. The results show that the FEJ-based VINS maintains consistency over time with NEES values near 3. Furthermore, the similarity in NEES values over time when comparing the proposed covariance estimator to the ceres method indicates that the covariance matrix computed using our algorithm is in close alignment with those produced by Ceres. Fig. 4 confirms that the estimation error remains within the bounds of $\pm 3\sigma$.

V. Conclusion and Future Work

In this paper, we have addressed one of the key challenges in optimization-based VINS: achieving efficient and consistent covariance recovery. Leveraging the FEJ methodology which fixes certain state linearization points when evaluating the Jacobian to improve the estimation performance, we have significantly accelerated the covariance estimation by avoiding redundant computation. The proposed approach not only ensures consistent state and covariance estimations but also achieves a speed that is 4-10 times faster than existing methods. We have rigorously designed our approach using the four predominant state-of-the-art marginalization methods, each with its unique FEJ prerequisites. Through comprehensive numerical studies, we have validated the consistency and efficiency of the proposed method, emphasizing the significant impact of the FEJ methodology in the evolution of VINS technology. In the future, we are interested in taking the benefits of FEJ to further accelerate the optimization process and downstream applications.

REFERENCES

[1] C. Chen, P. Geneva, Y. Peng, W. Lee, and G. Huang, "Optimization-based vins: Consistency, marginalization, and fej," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[2] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[3] T. Özaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1740–1747, 2017.

[4] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs," in *Proc. of the IEEE International Conference on Robotics and Automation*, Seattle, USA, 2015.

[5] C. Chen, Y. Yang, P. Geneva, W. Lee, and G. Huang, "Visual-inertial-aided online mav system identification," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyoto, Japan., 2022.

[6] J. Eisele, Z. Song, K. Nelson, and K. Mohseni, "Visual-inertial guidance with a plenoptic camera for autonomous underwater vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2777–2784, 2019.

[7] P. Geneva, N. Merrill, Y. Yang, C. Chen, W. Lee, and G. Huang, "Versatile 3d multi-sensor fusion for lightweight 2d localization," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

[8] C. Chen, P. Geneva, Y. Peng, W. Lee, and G. Huang, "Monocular visual-inertial odometry with planar regularities," in *Proc. of the IEEE International Conference on Robotics and Automation*, London, UK., 2023.

[9] D. S. Bayard, D. T. Conway, R. Brockers, J. H. Delaune, L. H. Matthies, H. F. Grip, G. B. Merewether, T. L. Brown, and A. M. San Martin, "Vision-based navigation for the nasa mars helicopter," in *AIAA Scitech 2019 Forum*, 2019, p. 1411.

[10] K. Wu, A. M. Ahmed, G. A. Georgiou, and S. 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.

[11] D. G. Kottas and S. 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.

[12] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[13] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Camera-imu-based localization: Observability analysis and consistency improvement," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 182–201, 2014.

[14] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[15] C. Chen, Y. Yang, P. Geneva, and G. Huang, "FEJ2: A consistent visual-inertial state estimator design," in *International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, 2022.

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

[17] T. Ke, K. J. Wu, and S. 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.

[18] Y. Peng, C. Chen, and G. Huang, "Ultrafast square-root filter-based VINS," in *Proc. International Conference on Robotics and Automation*, Yokohama, Japan, May 2024.

[19] ——, "Quantized visual-inertial odometry," in *Proc. International Conference on Robotics and Automation*, Yokohama, Japan, May 2024.

[20] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[21] S. Leutenegger, "Okvis2: Realtime scalable visual-inertial slam with loop closure," *arXiv preprint arXiv:2202.09199*, 2022.

[22] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[23] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2019.

[24] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[25] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robotics and autonomous systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

[26] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 890–897, 2001.

[27] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2916–2923.

[28] R. Valencia, M. Morta, J. Andrade-Cetto, and J. M. Porta, "Planning reliable paths with pose slam," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 1050–1059, 2013.

[29] P. Whaite and F. P. Ferrie, "Autonomous exploration: Driven by uncertainty," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 193–205, 1997.

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

[31] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph slam," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5748–5755.

[32] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual slam using a reduced pose graph," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 54–61.

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

[34] A. Davison and D. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[35] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[36] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2013.

[37] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[38] T.-C. Dong-Si and A. I. Mourikis, "Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5655–5662.

[39] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[40] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter, "Visually mapping the rms titanic: Conservative covariance estimates for slam information filters," *The international journal of robotics research*, vol. 25, no. 12, pp. 1223–1242, 2006.

[41] R. Eustice, H. Singh, J. J. Leonard, M. R. Walter, and R. Ballard, "Visually navigating the rms titanic with slam information filters." in *Robotics: Science and Systems*, vol. 2005, 2005, pp. 57–64.

[42] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[43] G. H. Golub and R. J. Plemmons, "Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition," *Linear Algebra and Its Applications*, vol. 34, pp. 3–28, 1980.

[44] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik, "Fast covariance recovery in incremental nonlinear least square solvers," in *2015 IEEE*

International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 4636–4643.

[45] V. Ila, L. Polok, M. Solony, and P. Svoboda, "Slam++-a highly efficient and temporally scalable incremental slam framework," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017.

[46] C. Chen, P. Geneva, Y. Peng, W. Lee, and G. Huang, "Technical report: Optimization-based VINS: Consistency, marginalization, and fej," University of Delaware, Tech. Rep. RPNG-2023-GRAPH, 2023. [Online]. Available: https://udel.edu/~ghuang/papers/tr_graph.pdf

[47] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., Mar. 2005.

[48] C. Hertzberg, R. Wagner, U. Frese, and L. SchröDer, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, Jan. 2013.

[49] C. Chen, Y. Peng, and G. Huang, "Technical report: Fast and consistent covariance recovery for sliding-window optimization-based vins," University of Delaware, Tech. Rep. RPNG-2024-COV, 2024. [Online]. Available: https://udel.edu/~ghuang/papers/tr_cov.pdf

[50] P. Geneva, K. Eckenhoff, and G. Huang, "A linear-complexity EKF for visual-inertial navigation with loop closures," in *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[51] K. Eckenhoff, P. Geneva, and G. Huang, "Closed-form preintegration methods for graph-based visual-inertial navigation," *International Journal of Robotics Research*, vol. 38, no. 5, pp. 563–586, 2019. [Online]. Available: https://github.com/rpng/cpi

[52] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," https://github.com/ceres-solver/ceres-solver, 2022.