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

Chuchu Chen - ccchu@udel.edu
Yuxiang Peng - yxpeng@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-2023-COV
Last Updated - September 28, 2023

Contents

1 Optimization-Based VINS

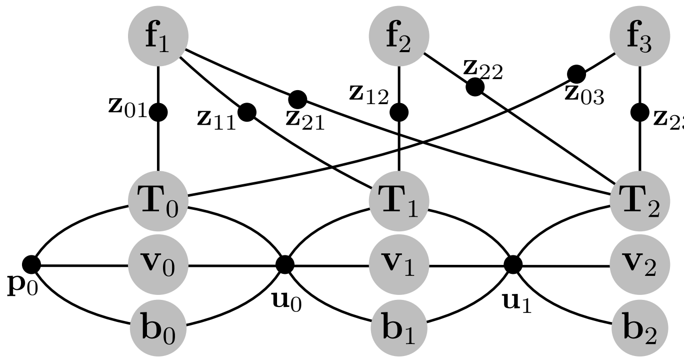


Figure 1: Example visual-inertial factor graph of bearing observations $\mathbf{z}$, inertial readings $\mathbf{u}$, and prior $\mathbf{p}_0$. The nodes (state variables) are represented as grey circles and edges (measurements) connect their related states. $\mathbf{f}_j$ denotes the jth feature, $\mathbf{T}_i$ is robot pose at t_i, $\mathbf{v}_i$ and $\mathbf{b}_i$ are the robot velocity and biases.

We formulate the nonlinear least squares (NLS) problem over the entire trajectory up to the current time t_k. The system state consists of the current navigation states, $\mathbf{x}_k$, and 3D features, $\mathbf{x}_f$:

$$\mathbf{x}_{0:k} = \begin{bmatrix} \mathbf{x}_0^\top & \dots & \mathbf{x}_k^\top & \mathbf{x}_f^\top \end{bmatrix}^\top \tag{1}$$

$$\mathbf{x}_k = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_k}^\top & {}^{G}\mathbf{v}_{I_k}^\top & \mathbf{b}_{g,k}^\top & \mathbf{b}_{a,k}^\top \end{bmatrix}^\top \tag{2}$$

$$\mathbf{x}_f = \begin{bmatrix} {}^{G}\mathbf{f}_1{}^\top & \dots & {}^{G}\mathbf{f}_g^\top \end{bmatrix}^\top \tag{3}$$

where ${}^{I}_{G}\bar{q}$ is the unit quaternionthat represents the rotation ${}^{I}_{G}\mathbf{R}$ from global frame $\{G\}$ to the IMU frame $\{I\}$. Note that 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 [1] 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 [2]. ${}^{G}\mathbf{p}_I$ 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. A common visualization technique in the optimization-based method is the "factor graph" [see Figure 1] for which states are represented as graph nodes and measurements are represented as edges which connect to their involve states.

1.1 Consistent FEJ-based VINS

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 maximizing the posterior pdf leveraging: 1) prior information $\mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0)$, 2) IMU motion constraints $\mathbf{u}$, and 3) camera observation measurements $\mathbf{z}$:

$$p(\mathbf{x}_{0:k}|\mathcal{Z}_{0:k}) \propto p(\mathbf{x}_0) \prod_{i=0}^{k-1} p(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i) \prod_{\mathbf{z}_{ij} \in \mathcal{Z}_{0:k}} p(\mathbf{z}_{ij}|\mathbf{x}_i, \mathbf{f}_j) \tag{4}$$

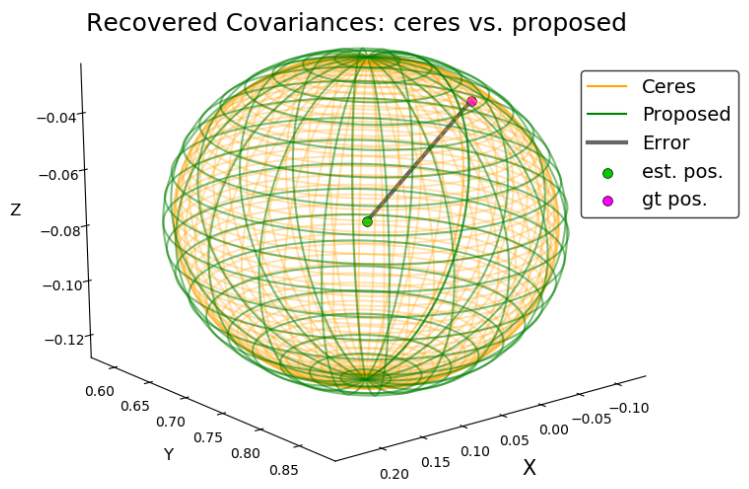


Figure 2: An example of robot position covariance recovered by Ceres and proposed methods. Green and pink dots are the estimated and groundtruth positions, respectively. The black line shows the estimation error.

where the set $\mathcal{Z}_{0:k}$ denotes all measurements between $[t_0, t_k]$. Under Gaussian distribution assumption, this pdf can be written as:

$$p(\mathbf{x}_{0:k}|\mathcal{Z}_{0:k}) \propto \frac{1}{\sqrt{(2\pi)^n|\mathbf{P}_0|}} \exp\left(-\mathcal{C}_{p_0}\right) \prod_{i=0}^{k-1} \frac{1}{\sqrt{(2\pi)^d|\mathbf{Q}_i|}} \exp(-\mathcal{C}_{I_i}) \prod_{\mathbf{z}_{ij}\in\mathcal{Z}_{0:k}} \frac{1}{\sqrt{(2\pi)^m|\mathbf{R}_{ij}|}} \exp(-\mathcal{C}_{f_{ij}})$$

where n is the dimension of prior state $\mathbf{x}_0$, d denote the size of robot state $\mathbf{x}_k$ and m is the dimension of measurement $\mathbf{z}_{ij}$. Maximizing the above pdf is equivalent to minimizing:

$$\mathcal{C}(\mathbf{x}_{0:k}) = \mathcal{C}_{p_0} + \sum_{i=0}^{k-1} \mathcal{C}_{I_i} + \sum_{\mathbf{z}_{ij}\in\mathcal{Z}_{0:k}} \mathcal{C}_{f_{ij}} \tag{5}$$

where we define the following costs:

$$\text{Prior:} \quad \mathcal{C}_{p_0} = \frac{1}{2}||\mathbf{x}_0 \boxminus \hat{\mathbf{x}}_0||^2_{\mathbf{P}_0} \tag{6}$$

$$\text{Inertial: [3]} \quad \mathcal{C}_{I_i} = \frac{1}{2}||\mathbf{x}_{i+1} \boxminus \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)||^2_{\mathbf{Q}_i} \tag{7}$$

$$\text{Camera: [4]} \quad \mathcal{C}_{f_{ij}} = \frac{1}{2}||\mathbf{z}_{ij} \boxminus \mathbf{h}(\mathbf{x}_i, \mathbf{f}_j)||^2_{\mathbf{R}_{ij}} \tag{8}$$

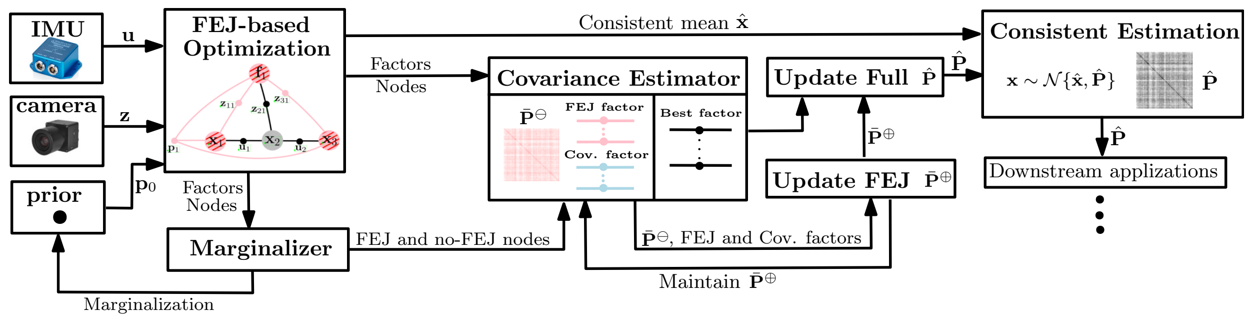
where $||\mathbf{a}||^2_{\mathbf{W}} := \mathbf{a}^\top\mathbf{W}^{-1}\mathbf{a}$ and can be solved iteratively given an initial linearization point. The second-order Taylor series of the l-th iteration with linearization point $\hat{\mathbf{x}}^l_{0:k}$ is:

$$\mathcal{C}(\hat{\mathbf{x}}^l_{0:k} \boxplus \delta\mathbf{x}^l_{0:k}) \simeq \mathcal{C}(\hat{\mathbf{x}}^l_{0:k}) + \mathbf{b}^{l^\top}\delta\mathbf{x}^l_{0:k} + \frac{1}{2}\delta\mathbf{x}^{l^\top}_{0:k}\mathbf{A}^l\delta\mathbf{x}^l_{0:k} \tag{9}$$

where $\mathbf{b}^l$ and $\mathbf{A}^l$ are the linearized gradient and Hessian of the cost function, respectively.

$$\mathbf{b}^l = \mathbf{\Gamma}_0^\top\mathbf{P}_0^{-1}(\hat{\mathbf{x}}^l_0 \boxminus \hat{\mathbf{x}}_0) + \sum_{i=0}^{k-1} \mathbf{\Phi}_i^{l^\top}\mathbf{Q}_i^{-1}\left(\hat{\mathbf{x}}^l_{i+1} \boxminus \mathbf{f}(\hat{\mathbf{x}}^l_i, \mathbf{u}_i)\right) + \sum_{\mathbf{z}_{i,j}\in\mathcal{Z}_{0:k}} \mathbf{H}_{ij}^{l^\top}\mathbf{R}_{ij}^{-1}\left(\mathbf{z}_{ij} \boxminus \mathbf{h}(\hat{\mathbf{x}}^l_{0:k})\right) \tag{10}$$

$$\mathbf{A}^l = \mathbf{\Gamma}_0^\top\mathbf{P}_0^{-1}\mathbf{\Gamma}_0 + \sum_{i=0}^{k-1} \mathbf{\Phi}_i^{l^\top}\mathbf{Q}_i^{-1}\mathbf{\Phi}_i^l + \sum_{\mathbf{z}_{ij}\in\mathcal{Z}_{0:k}} \mathbf{H}_{ij}^{l^\top}\mathbf{R}_{ij}^{-1}\mathbf{H}_{ij}^l \tag{11}$$



IMU
camera
prior
u
z
$\mathbf{p}_0$
FEJ-based Optimization
Factors
Nodes
Marginalizer
Marginalization
Covariance Estimator
FEJ factor
Cov. factor
Best factor
Consistent mean $\hat{\mathbf{x}}$
Update Full $\bar{\mathbf{P}}$
Update FEJ $\bar{\mathbf{P}}^{\oplus}$
FEJ and no-FEJ nodes
$\bar{\mathbf{P}}^{\ominus}$, FEJ and Cov. factors
Maintain $\bar{\mathbf{P}}^{\oplus}$
Consistent Estimation
$\mathbf{x} \sim \mathcal{N}\{\hat{\mathbf{x}}, \hat{\mathbf{P}}\}$
Downstream applications

Figure 3: Diagram of FEJ-based VINS within nonlinear optimization with efficient and consistent covariance recovery.

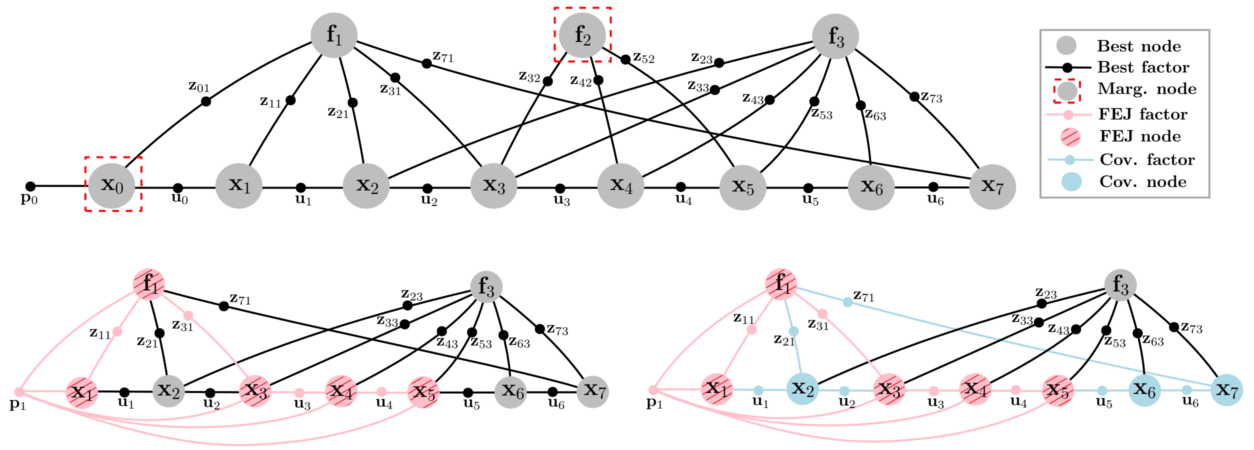


Figure 4: **Top**: Example visual-inertial factor graph: Grey circles represent nodes (states) with edges (measurements) connecting related states. $\mathbf{f}_j$ is the jth feature, $\mathbf{x}_i$ is robot state at t_i. **Bottom left**: the resulting graph after marginalizing $\mathbf{x}_0$, $\mathbf{f}_2$, FEJ node with fixed linearization points are shaded in pink, while pink edges denote those evaluated and linearized with FEJ. **Bottom right**: node and factors used to recover the FEJ covariance. Blue are the extra "Cov." factor and nodes used to recover the FEJ covariance, in addition to the FEJ (pink) ones.

where $\mathbf{\Gamma}_0 = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$ with $n = \mathtt{dim}(\mathbf{x}_0)$ is the size of initial state $\mathbf{x}_0$. This indicates that we add a prior factor to the initial state variables up to the time t_k. The correction term and the updated state $\hat{\mathbf{x}}_{0:k}^{l+1}$ can be solved by:

$$\mathbf{A}^l \delta \mathbf{x}_{0:k}^l = -\mathbf{b}^l \quad \Rightarrow \quad \hat{\mathbf{x}}_{0:k}^{l+1} = \hat{\mathbf{x}}_{0:k}^l \boxplus \delta \mathbf{x}_{0:k}^l \tag{12}$$

Given initial state $\hat{\mathbf{x}}_0$, this iterative algorithm will compute the global minimal estimates (MAP) for the entire state $\mathbf{x}_{0:k}$ given all available measurements within time period $[t_0, t_k]$.

2 Efficient Covariance Estimate

Recovering the estimated covariance while solving the NLS [Eq. (5)] in real-time can be a computational bottleneck due to the need to invert the information matrix $\mathbf{A}$, such that,

$$\mathbf{P} = \mathbf{A}^{-1} \tag{13}$$

Naively performing matrix inversion will be cubic complexity, $O(n^3)$, where n is the dimension of states. In practice, performing this operation with every new measurement becomes prohibitively expensive.

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 2.1. Next, we present how to recover covariance using IMU and camera factors in Section 2.2. Section 2.3 explains in detail the proposed fast covariance recovery algorithm in conjunction with various marginalization methods.

2.1 FEJ-based Covariance Recovery

When solving the NLS problem with FEJ method, some measurement functions and their corresponding Hessian matrices will be evaluated using fixed linearization points (i.e., using 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 4 (bottom left), the FEJ factors (colored in pink) and the corresponding covariances do not require re-evaluation in subsequent instances. Drawing from these insights, we first recover and maintain a "FEJ" covariance $\bar{\mathbf{P}}$, use the FEJ measurements:

$$\bar{\mathbf{P}}^{\oplus} = \Delta\mathbf{P}(\bar{\mathbf{P}}^{\ominus}, \bar{\mathbf{H}}, \bar{\mathbf{R}}) \tag{14}$$

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, its linearized Jacobian $\hat{\mathbf{H}}$ and corresponding noise matrix $\hat{\mathbf{R}}$ as:

$$\hat{\mathbf{P}}^{\oplus} = \Delta\mathbf{P}(\bar{\mathbf{P}}^{\oplus}, \hat{\mathbf{H}}, \hat{\mathbf{R}}) \tag{15}$$

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

Given the cost functions, we first show the corresponding linearized covariance matrix. In what follows, we show one simple example, where the state vector is defined from two timestamps as $\mathbf{x} = [\mathbf{x}_k^\top, \mathbf{x}_{k+1}^\top]^\top$ The cost function of the optimization problem can be formulated as:

$$\mathcal{C}(\mathbf{x}_k, \mathbf{x}_{k+1}) = \mathcal{C}_{p_k} + \mathcal{C}_{I_k} + \mathcal{C}_{C_{k+1}} \tag{16}$$

where $\mathcal{C}_{p_k}$ is the prior factor that connect to $\mathbf{x}_k$, $mathcalC_{I_k}$ and $\mathcal{C}_{C_{k+1}}$ are IMU preintegration and camera measurement factor, respectively. The linearized cost can be derived as:

$$\frac{1}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2_{\mathbf{P}_k} + \frac{1}{2}\left\| \begin{bmatrix}\mathbf{I} & -\mathbf{\Phi}_k^\top\end{bmatrix} \begin{bmatrix}\mathbf{x}_{k+1} \\ \mathbf{x}_k\end{bmatrix} \right\|^2_{\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^\top} + \|\tilde{\mathbf{z}}_{k+1} - \mathbf{H}_{k+1}\tilde{\mathbf{x}}_{k+1}\|^2_{\mathbf{R}_{k+1}} \tag{17}$$

Omitting some derivations we are able to get the information matrix with respect to the states as:

$$\begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_k\end{bmatrix}^{-1} + \begin{bmatrix}\mathbf{I} \\ -\mathbf{\Phi}_k^\top\end{bmatrix} \left(\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^\top\right)^{-1} \begin{bmatrix}\mathbf{I} & -\mathbf{\Phi}_k\end{bmatrix} + \begin{bmatrix}-\mathbf{H}_{k+1}^\top \\ \mathbf{0}\end{bmatrix} \mathbf{R}_{k+1}^{-1} \begin{bmatrix}-\mathbf{H}_{k+1} & \mathbf{0}\end{bmatrix} \tag{18}$$

2.2 Covariance Update Methods

We now explain how to derive $\mathbf{\Delta P}(\cdot)$ with IMU and camera measurements. For clarity, we use a versatile notation without differentiating between FEJ ($\bar{\mathbf{P}}$) and full covariance ($\hat{\mathbf{P}}$) as the approaches are applicable for both.

2.2.1 IMU preintegration factor

Given the linearized IMU preintegration measurement:

$$\tilde{\mathbf{u}} \simeq -\boldsymbol{\Phi}_k \tilde{\mathbf{x}}_k + \tilde{\mathbf{x}}_{k+1} + \mathbf{w}_k \tag{19}$$

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

$$\mathbf{P}^{\oplus} = \begin{bmatrix} \boldsymbol{\Phi}_k \mathbf{P}^{\ominus} \boldsymbol{\Phi}_k^{\top} + \mathbf{Q}_k & \boldsymbol{\Phi}_k \mathbf{P}^{\ominus} \\ \mathbf{P}^{\ominus} \boldsymbol{\Phi}_k^{\top} & \mathbf{P}^{\ominus} \end{bmatrix} \tag{20}$$

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

Lemma 2.1. *Given the same linearization point, the propagated and augmented covariance matrix, Eq. (20), is equivalent to inverting the information matrix.*

Proof. We first consider the covariance after adding prior and IMU cost to the system:

$$\begin{aligned} \mathbf{P}_{k+1|k} &= \left(\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_k^{-1} \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ -\boldsymbol{\Phi}_k^{\top} \end{bmatrix} \left(\mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} \right)^{-1} \begin{bmatrix} \mathbf{I} & -\boldsymbol{\Phi}_k \end{bmatrix} \right)^{-1} & (21) \\ &= \left(\begin{bmatrix} \left(\mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} \right)^{-1} & -\left(\mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} \right)^{-1} \boldsymbol{\Phi}_k \\ -\boldsymbol{\Phi}_k^{\top} \left(\mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} \right)^{-1} & \mathbf{P}_k^{-1} - \boldsymbol{\Phi}_k^{\top} \left(\mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} \right)^{-1} \boldsymbol{\Phi}_k \end{bmatrix} \right)^{-1} & (22) \\ &= \begin{bmatrix} \boldsymbol{\Phi}_k \mathbf{P}_k \boldsymbol{\Phi}_k^{\top} + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} & \boldsymbol{\Phi}_k \mathbf{P}_k \\ \mathbf{P}_k \boldsymbol{\Phi}_k^{\top} & \mathbf{P}_k \end{bmatrix} & (23) \end{aligned}$$

where we have leveraged the partitioned matrix inversion equation.

In what follows, we prove that if the linearization point for evaluating Jacobians (i.e., $\boldsymbol{\Phi}_k$ and $\mathbf{G}_k$) remains the same, the proposed method will result in the same covariance for the prior and IMU cost terms. Given the prior state and covariance, the state augmentation and covariance propagation can be derived by:

$$\begin{aligned} \mathbf{P}_{k+1|k} &= \begin{bmatrix} \boldsymbol{\Phi}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}_k & \mathbf{P}_k \\ \mathbf{P}_k & \mathbf{P}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{\top} + \begin{bmatrix} \mathbf{G}_k \\ \mathbf{0} \end{bmatrix} \mathbf{Q}_k \begin{bmatrix} \mathbf{G}_k^{\top} & \mathbf{0} \end{bmatrix} & (24) \\ &= \begin{bmatrix} \boldsymbol{\Phi}_k \mathbf{P}_k \boldsymbol{\Phi}_k^{\top} + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^{\top} & \boldsymbol{\Phi}_k \mathbf{P}_k \\ \mathbf{P}_k \boldsymbol{\Phi}_k^{\top} & \mathbf{P}_k \end{bmatrix} & (25) \end{aligned}$$

We thus finish the proof. □

2.2.2 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:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_x \tilde{\mathbf{x}} + \mathbf{H}_f \tilde{\mathbf{f}} + \mathbf{n} = \mathbf{H} \begin{bmatrix} \tilde{\mathbf{x}}^{\top} & \tilde{\mathbf{f}}^{\top} \end{bmatrix}^{\top} + \mathbf{n} \tag{26}$$

If the feature is already in the covariance, we can use Eq. (26) to perform an EKF update:

$$\mathbf{P}^{\oplus} = \mathbf{P}^{\ominus} - \mathbf{P}^{\ominus} \mathbf{H}^{\top} \left(\mathbf{H} \mathbf{P}^{\ominus} \mathbf{H}^{\top} + \mathbf{R} \right)^{-1} \mathbf{H}_x \mathbf{P}^{\ominus} \tag{27}$$

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.

2.2.2.1 Feature Initialization We first perform Givens rotations to $\mathbf{H}_f$ in Eq. (26) results in:

$$\begin{bmatrix} \tilde{\mathbf{z}}_{K1} \\ \tilde{\mathbf{z}}_{K2} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{x1} \\ \mathbf{H}_{x2} \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{H}_{f1} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{f}} + \begin{bmatrix} \mathbf{n}_{K1} \\ \mathbf{n}_{K2} \end{bmatrix} \tag{28}$$

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

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}^{\ominus} & \mathbf{P}_{xf} \\ \mathbf{P}_{xf}^{\top} & \mathbf{P}_{ff} \end{bmatrix} = \begin{bmatrix} \mathbf{P}^{\ominus} & -\mathbf{P}^{\ominus}\mathbf{H}_{x1}^{\top}\mathbf{H}_{f1}^{-\top} \\ \mathbf{P}_{xf}^{\top} & \mathbf{P}_{ff} \end{bmatrix} \tag{29}$$

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:

$$\mathbf{P}^{\oplus} = \mathbf{P} - \mathbf{P}\mathbf{H}_{x2}^{\top}\left(\mathbf{H}\mathbf{P}\mathbf{H}_{x2}^{\top} + \mathbf{R}_2\right)^{-1}\mathbf{H}_{x2}\mathbf{P} \tag{30}$$

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

2.2.2.2 MSCKF Feature One can remove the feature by projecting the linearized measurement function, Eq. (26), onto the left nullspace $\mathbf{N}$ of $\mathbf{H}_f$:

$$\mathbf{N}^{\top}\tilde{\mathbf{z}} \simeq \mathbf{N}^{\top}\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{N}^{\top}\mathbf{n} \;\Rightarrow\; \tilde{\mathbf{z}}' = \mathbf{H}_x'\tilde{\mathbf{x}} + \mathbf{n}' \tag{31}$$

We can then directly perform EKF update:

$$\mathbf{P}^{\oplus} = \mathbf{P}^{\ominus} - \mathbf{P}^{\ominus}\mathbf{H}_x'^{\top}\left(\mathbf{H}_x'\mathbf{P}^{\ominus}\mathbf{H}_x'^{\top} + \mathbf{R}'\right)^{-1}\mathbf{H}_x'\mathbf{P}^{\ominus} \tag{32}$$

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

Lemma 2.2. *Given the same linearized measurement equation, Eq. (26), the robot covariance will be the same if performing the above two methods in Section 2.2.2.1 and 2.2.2.2.*

Proof. We start with the MSCKF feature update Eq. (31). Given the linearized measurement equation, Eq.(26), the nullspace can be found using QR decomposition as follows:

$$\mathbf{H}_f = \mathbf{Q}\mathbf{R} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix} \tag{33}$$

Multiplying the linearized measurement equation by the nullspace of the feature Jacobian, $\mathbf{Q}_2$, from the left yields:

$$\mathbf{Q}_2^{\top}\tilde{\mathbf{z}} \simeq \mathbf{Q}_2^{\top}\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{Q}_2^{\top}\mathbf{n} \tag{34}$$

To maintain consistent notation throughout the proof, we use $\mathbf{Q}_2$ to denote the nullspace, which corresponds to $\mathbf{N}$ in Eq. (31). The updated covariance matrix, $\mathbf{P}^{\oplus}$, is derived by inverting the information matrix as follows:

$$\mathbf{P}^{\oplus} = \left[\mathbf{P}^{\ominus^{-1}} + (\mathbf{Q}_2^{\top}\mathbf{H}_x)^{\top}(\mathbf{Q}_2^{\top}\mathbf{R}\mathbf{Q}_2)^{-1}(\mathbf{Q}_2^{\top}\mathbf{H}_x)\right]^{-1} \tag{35}$$

We then look into the feature initialization and update approach introduced in Section 2.2.2.1. Similarly, the QR decomposing is applied to $\mathbf{H}_f$ and the residual $\tilde{\mathbf{z}}$ in Eq. (26), which will result in Eq. (28). In more detailed derivation, we have:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{H}_f\tilde{\mathbf{f}} + \mathbf{n} \tag{36}$$

$$\Rightarrow \quad \mathbf{Q}^\top\tilde{\mathbf{z}} = \mathbf{Q}^\top\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{Q}^\top\mathbf{H}_f\tilde{\mathbf{f}} + \mathbf{Q}^\top\mathbf{n} \tag{37}$$

$$\Rightarrow \quad \begin{bmatrix}\mathbf{Q}_1^\top\tilde{\mathbf{z}}\\ \mathbf{Q}_2^\top\tilde{\mathbf{z}}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}_1^\top\\ \mathbf{Q}_2^\top\end{bmatrix}\mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{Q}^\top\mathbf{Q}\mathbf{R}\tilde{\mathbf{f}} + \begin{bmatrix}\mathbf{Q}_1^\top\mathbf{n}\\ \mathbf{Q}_2^\top\mathbf{n}\end{bmatrix} \tag{38}$$

$$\Rightarrow \quad \begin{bmatrix}\mathbf{Q}_1^\top\tilde{\mathbf{z}}\\ \mathbf{Q}_2^\top\tilde{\mathbf{z}}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}_1^\top\mathbf{H}_x\\ \mathbf{Q}_2^\top\mathbf{H}_x\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}\mathbf{R}_1\\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{f}} + \begin{bmatrix}\mathbf{Q}_1^\top\mathbf{n}\\ \mathbf{Q}_2^\top\mathbf{n}\end{bmatrix} \tag{39}$$

$$\Rightarrow \quad \begin{bmatrix}\tilde{\mathbf{z}}_{K1}\\ \tilde{\mathbf{z}}_{K2}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{x1}\\ \mathbf{H}_{x2}\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}\mathbf{H}_{f1}\\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{f}} + \begin{bmatrix}\mathbf{n}_{K1}\\ \mathbf{n}_{K2}\end{bmatrix} \tag{40}$$

As such, using the upper linear system, the covariance, Eq. (29), can be derived as

$$\mathbf{P} = \begin{bmatrix}\mathbf{P}^\ominus & \mathbf{P}_{xf}\\ \mathbf{P}_{xf}^\top & \mathbf{P}_{ff}\end{bmatrix} \tag{41}$$

$$= \begin{bmatrix}\mathbf{P}^\ominus & -\mathbf{P}^\ominus(\mathbf{Q}_1^\top\mathbf{H}_x)^\top\mathbf{R}^{-\top}\\ \mathbf{P}_{xf}^\top & \mathbf{R}^{-1}(\mathbf{Q}_1^\top\mathbf{H}_x\mathbf{P}^\ominus\mathbf{H}_x^\top\mathbf{Q}_1 + \mathbf{Q}_1^\top\mathbf{R}\mathbf{Q}_1)\mathbf{R}^{-\top}\end{bmatrix} \tag{42}$$

$$= \begin{bmatrix}\mathbf{P}^\ominus & -\mathbf{P}^\ominus\mathbf{H}_{x1}^\top\mathbf{H}_{f1}^{-\top}\\ \mathbf{P}_{xf}^\top & \mathbf{H}_{f1}^{-1}(\mathbf{H}_{x1}\mathbf{P}^\ominus\mathbf{H}_{x1}^\top + \mathbf{R}_1)\mathbf{H}_{f1}^{-\top}\end{bmatrix} \tag{43}$$

Next, we use the bottom linear system in Eq. (29) to perform the update using Eq. (42). The updated covariance can be derived as:

$$\mathbf{P}^\oplus = \mathbf{P} - \mathbf{P}\begin{bmatrix}\mathbf{Q}_2^\top\mathbf{H}_x & \mathbf{0}\end{bmatrix}^\top\left(\begin{bmatrix}\mathbf{Q}_2^\top\mathbf{H}_x & \mathbf{0}\end{bmatrix}\mathbf{P}\begin{bmatrix}\mathbf{H}_x^\top\mathbf{Q}_2\\ \mathbf{0}\end{bmatrix} + \mathbf{Q}_2^\top\mathbf{R}\mathbf{Q}_2\right)^{-1}\begin{bmatrix}\mathbf{Q}_2^\top\mathbf{H}_x & \mathbf{0}\end{bmatrix}\mathbf{P} \tag{44}$$

$$= \mathbf{P}^\ominus - \mathbf{H}_x^\top\mathbf{Q}_2\left[\mathbf{Q}_2^\top(\mathbf{H}_x\mathbf{P}^\ominus\mathbf{H}_x^\top + \mathbf{R})\mathbf{Q}_2\right]^{-1}\mathbf{Q}_2^\top\mathbf{H}_x \tag{45}$$

$$= \mathbf{P}^\ominus - \mathbf{H}_x^\top\mathbf{Q}_2\mathbf{Q}_2^{-1}(\mathbf{H}_x\mathbf{P}^\ominus\mathbf{H}_x^\top + \mathbf{R})^{-1}\mathbf{Q}_2^{-\top}\mathbf{Q}_2^\top\mathbf{H}_x \tag{46}$$

Leveraging the matrix inversion Lemma, Eq. (46) can be written as:

$$\mathbf{P}^\oplus = \left[\mathbf{P}^{\ominus^{-1}} + (\mathbf{Q}_2^\top\mathbf{H}_x)^\top(\mathbf{Q}_2^\top\mathbf{R}\mathbf{Q}_2)^{-1}(\mathbf{Q}_2^\top\mathbf{H}_x)\right]^{-1} \tag{47}$$

Given that the updated covariance derived from the MSCKF feature update method (as shown in Eq.(35)) matches that obtained from the feature initialization and update (as presented in Eq.(47)), our proof is complete. □

While it is feasible to use the aforementioned “MSCKF-like” method for incremental covariance update. In practice, when there is a new measurement, it requires full covariance recovery from scratch since all the state estimates change, which is computationally expensive. 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.

```
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. (5), (12)].
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. (20)].
      – Camera factor: select ones connect to FEJ features
          ∗ KEEP FEJ feature: If is not in $\bar{\mathbf{P}}^{\ominus}$, initialize feature
            [Eq. (28),(29),(30)]; otherwise, EKF update.
          ∗ Other FEJ feature: MSCKF update [Eq.(31)]
  • 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.(31)].
```

2.3 Algorithm

Fig. 3 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 [5, 6].

The optimal approach would recover 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. 4 (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 invertable, preventing us from obtaining the associated covariance.

Interestingly, our previous work found that applying FEJ to the IMU factors minimally affects estimation performance (see Table III in [5]). 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. 4 (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:

Table 1: Simulation parameters and prior standard deviations for measurement perturbations.

Parameter	Value	Parameter	Value
Gyro. White Noise	1.6968e-4	Gyro. Rand. Walk	1.9393e-5
Accel. White Noise	2.0000e-3	Accel. Rand. Walk	3.0000e-3
Cam Freq. (Hz)	10	IMU Freq. (Hz)	400
Num. Clones	10	Tracked Feat.	100
Min. Track Length	5	Max. SLAM Feat.	35

- **KEEP FEJ feature**: If a feature is FEJ'ed with the KEEP method (see Fig. 1(a) in [5]). We will initialize it, [See Section 2.2.2.1] 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.(31).

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. 4 (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. This can support downstream applications or evaluate the estimator's consistency.

3 Numerical Study

In line with our previous study [5], we simulate the realistic indoor dataset to test the proposed algorithm. We leverage the OpenVINS simulator [7], CPI preintegration [3] with the Ceres Solver [8] (see Fig. 2 in [5]and Table 1). In the following section, CKLAM shifts 2 robot states, others marginalize 1 each time.

3.1 Time evaluation

We first investigate the computational cost, comparing the proposed covariance recovery method to the Ceres Solver. While the Ceres solver recovers covariances either using dense SVD or sparse QR decomposition of the information matrix followed by a back substitution, we focus on the sparse QR approach since dense SVD is prohibitively slow. Shown in Fig. 5, we compared different covariance recovery algorithms (line colors), considering variances in sliding window sizes (y-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 IMU states in the sliding window, while "proposed" is our proposed FEJ-based method. Given space constraints, we display the runtime for each algorithm only with a feature size of 100. [1].

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

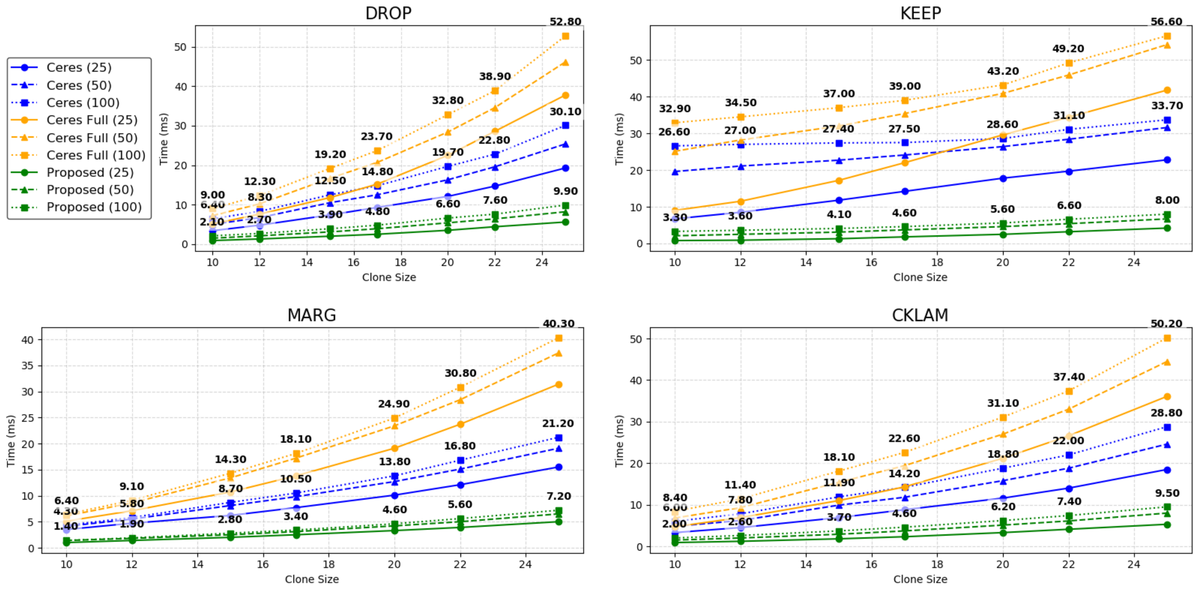


Figure 5: Timing comparison in covariance recovery across algorithms with varied marginalization methods. Line colors denote the different algorithms. "**Ceres (blue)**": utilizes Ceres to recover covariance of the most recent IMU pose (6 DoF), "**Ceres full (orange)**": use Ceres to recover all IMU 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). y-axis is the clone (sliding window) size, while x-axis is the computional time (ms).

Table 2: Timing comparison in covariance recovery across algorithms with different number of features and clone sizes, KEEP marginalization method.

Feat No.	**25**			**50**			**100**		
Clone size	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**
10	9.0	6.7	0.8	25.1	19.6	2.1	32.9	26.6	3.3
12	11.5	8.5	0.9	28.2	21.1	2.5	34.5	27.0	3.6
15	17.2	11.8	1.3	32.0	22.7	3.1	37.0	27.4	4.1
17	22.0	14.2	1.8	35.4	24.1	3.7	39.0	27.5	4.6
20	29.6	17.8	2.5	40.9	26.4	4.6	43.2	28.6	5.6
22	34.4	19.7	3.2	45.9	28.4	5.4	49.2	31.1	6.6
25	41.8	22.8	4.2	54.2	31.6	6.7	56.6	33.7	8.0

Table 3: Timing comparison in covariance recovery across algorithms with different number of features and clone sizes, DROP marginalization method.

Feat No.	**25**			**50**			**100**		
Clone size	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**
10	5.1	3.4	0.9	7.2	5.0	1.5	9.0	6.4	2.1
12	7.6	4.8	1.3	10.2	6.7	2.1	12.3	8.3	2.7
15	11.8	7.4	2.0	16.7	10.5	3.1	19.2	12.5	3.9
17	15.2	9.3	2.5	20.7	12.5	3.9	23.7	14.8	4.8
20	22.5	12.1	3.5	28.4	16.3	5.4	32.8	19.7	6.6
22	28.6	14.7	4.4	34.6	19.6	6.4	38.9	22.8	7.6
25	37.8	19.3	5.6	46.2	25.4	8.2	52.8	30.1	9.9

Table 4: Timing comparison in covariance recovery across algorithms with different number of features and clone sizes, MARG marginalization method.

Feat No.		**25**			**50**			**100**	
Clone size	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**
10	5.1	3.5	1.0	6.1	4.1	1.4	6.4	4.3	1.4
12	7.1	4.7	1.4	8.6	5.5	1.8	9.1	5.8	1.9
15	10.7	6.2	2.0	13.4	8.1	2.5	14.3	8.7	2.8
17	13.8	7.7	2.5	17.2	9.8	3.1	18.1	10.5	3.4
20	19.1	10.1	3.3	23.4	12.7	4.2	24.9	13.8	4.6
22	23.7	12.1	3.9	28.4	15.1	5.0	30.8	16.8	5.6
25	31.4	15.5	5.0	37.4	19.1	6.5	40.3	21.2	7.2

Table 5: Timing comparison in covariance recovery across algorithms with different number of features and clone sizes, CKLAM marginalization method.

Feat No.		**25**			**50**			**100**	
Clone size	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**	**Ceres Full**	**Ceres**	**Proposed**
10	4.7	3.3	0.9	6.8	4.8	1.5	8.4	6.0	2.0
12	6.9	4.5	1.2	9.3	6.2	2.0	11.4	7.8	2.6
15	11.0	6.9	1.8	15.4	9.9	2.9	18.1	11.9	3.7
17	14.3	8.8	2.3	19.4	11.8	3.7	22.6	14.2	4.6
20	21.2	11.6	3.3	27.0	15.8	5.1	31.1	18.8	6.2
22	26.6	14.0	4.1	33.0	18.8	6.1	37.4	22.0	7.4
25	36.1	18.5	5.3	44.5	24.6	8.0	50.2	28.8	9.5

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, while Ceres focuses only on the latest 6DoF pose, which demonstrates potential advantages in real-world applications. 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. Figure 6 further demonstrates the superior efficiency of our proposed algorithm.

Fig. 9 (left) displays total, FEJ state sizes, and states for covariance recovery (pink and blue nodes in Fig. 4) for KEEP method when the number of features is 100. As discussed in Section 2.3, 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.

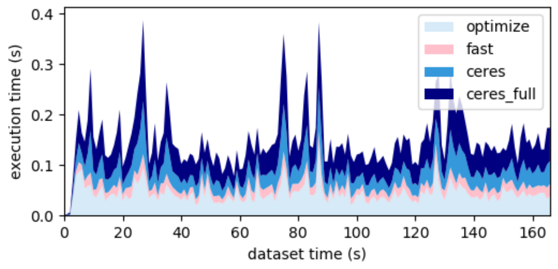


Figure 6: Timing of the optimization and covariance recovery with different algorithms.

Table 6: Average ATE and NEES (20 runs)

Marg. method	ATE (deg/m)	NEES-ceres	NEES-proposed
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

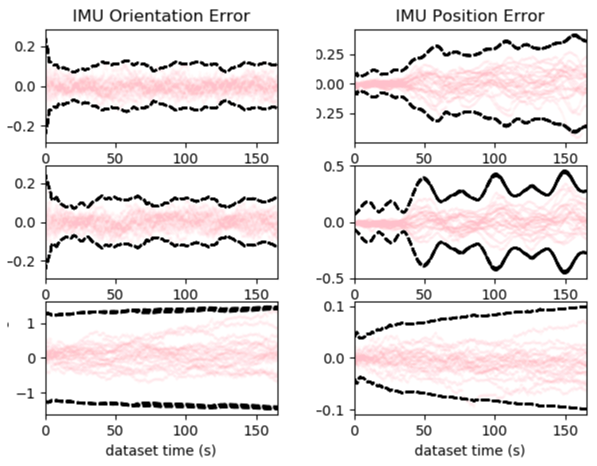


Figure 7: IMU x position and roll angle errors (pink) with $\pm 3\sigma$ bounds (dashed black) across 20 Monte Carlo runs (KEEP).

3.2 Consistency Evaluation

We then look into the consistency of the system and the recovered covariance. In this section, we present simulation 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 6 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 2.3. The KEEP method, due to its longer feature tracking, unsurprisingly achieves the most accurate performance with the smallest ATE. Fig. 8 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 strongly indicates that the covariance matrix computed using our algorithm is in close alignment with those produced by Ceres. Fig. 7 confirms that the estimation error remains consistent within the bounds of $\pm 3\sigma$.

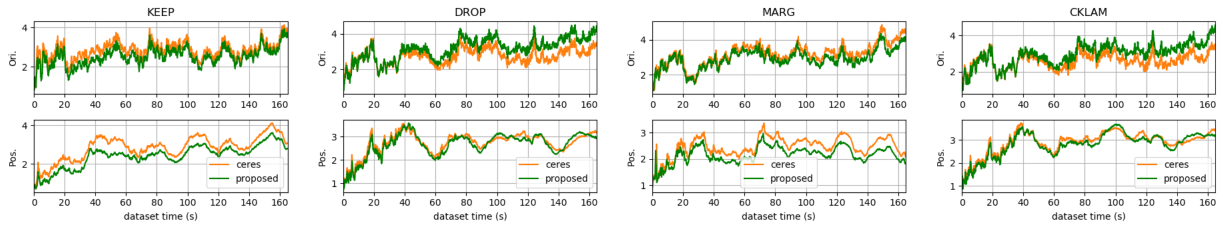


Figure 8: NEES for IMU orientation & position over 20 runs: Ceres vs. proposed

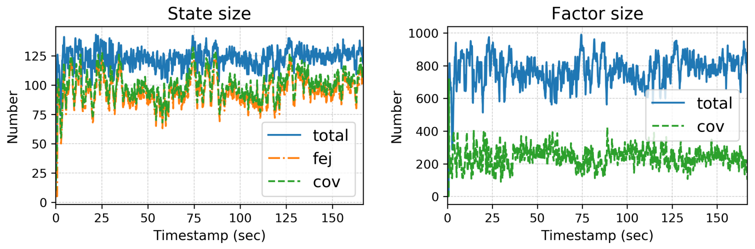


Figure 9: The size of states (left) and factors (right). "total" represents the total size, "fej" is the number of FEJ nodes, and "cov" denotes the number of the states and factors we used to recover the covariance.

References

[1] Nikolas Trawny and Stergios I. Roumeliotis. *Indirect Kalman Filter for 3D Attitude Estimation*. Tech. rep. University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.

[2] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz SchröDer. "Integrating Generic Sensor Fusion Algorithms with Sound State Representations Through Encapsulation of Manifolds". In: *Information Fusion* 14.1 (Jan. 2013), pp. 57–77. ISSN: 1566-2535.

[3] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. "Closed-form Preintegration Methods for Graph-based Visual-Inertial Navigation". In: *International Journal of Robotics Research* 38.5 (2019), pp. 563–586.

[4] 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.

[5] 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.

[6] Chuchu Chen, Patrick Geneva, Yuxiang Peng, Woosik Lee, and Guoquan Huang. *Technical Report: Optimization-based VINS: Consistency, Marginalization, and FEJ*. Tech. rep. RPNG-2023-GRAPH. University of Delaware, 2023. URL: https://udel.edu/~ghuang/papers/tr_graph.pdf.

[7] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. "A Linear-Complexity EKF for Visual-Inertial Navigation with Loop Closures". In: *Proc. International Conference on Robotics and Automation*. Montreal, Canada, May 2019.

[8] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. https://github.com/ceres-solver/ceres-solver. 2022.