sensors

MDPI

Article

Online IMU Self-Calibration for Visual-Inertial Systems

Yao Xiao [1], Xiaogang Ruan [1], Jie Chai [1], Xiaoping Zhang [2] and Xiaoqing Zhu [1,*]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; xiaoyao1103@emails.bjut.edu.cn (Y.X.); adrxg@bjut.edu.cn (X.R.); chaijie@emails.bjut.edu.cn (J.C.)
[2] College of Electrical and Control Engineering, North China University of Technology, Beijing 100144, China; zhangxiaoping369@163.com
* Correspondence: alex.zhuxq@bjut.edu.cn

Received: 20 February 2019; Accepted: 31 March 2019; Published: 4 April 2019

Abstract: Low-cost microelectro mechanical systems (MEMS)-based inertial measurement unit (IMU) measurements are usually affected by inaccurate scale factors, axis misalignments, and g-sensitivity errors. These errors may significantly influence the performance of visual-inertial methods. In this paper, we propose an online IMU self-calibration method for visual-inertial systems equipped with a low-cost inertial sensor. The goal of our method is to concurrently perform 3D pose estimation and online IMU calibration based on optimization methods in unknown environments without any external equipment. To achieve this goal, we firstly develop a novel preintegration method that can handle the IMU intrinsic parameters error propagation. Then, we frame IMU calibration problem into general factors so that we can easily integrate the factors into the current graph-based visual-inertial frameworks and jointly optimize the IMU intrinsic parameters as well as the system states in a big bundle. We evaluate the proposed method with a publicly available dataset. Experimental results verify that the proposed approach is able to accurately calibrate all the considered parameters in real time, leading to significant improvement of estimation precision of visual-inertial system (VINS) compared with the estimation results with offline precalibrated IMU measurements.

Keywords: visual-inertial system; visual odometry; SLAM; IMU calibration; sensor fusion

1. Introduction

Vision-based motion estimation and 3D reconstruction is a very active field of research in robotics and computer vision communities over the last decades due to its potential applications, such as robot navigation, autonomous driving, augmented reality (AR) and virtual reality (VR). Different sensor setups can be used for vision-based motion estimation: monocular [1,2], stereo [3,4], RGB-D [5,6] and visual-inertial [7,8]. Among these, the combination of cameras and inertial sensors, also known as visual-inertial system (VINS), has attracted more and more attention in recent years [7–21]. On the one hand, camera and inertial measurements offer complementary nature, and the combination of these two sensors can lead to significant improvements in accuracy of the estimation results and robustness of the system. On the other hand, with the development of microelectro mechanical systems (MEMS) inertial sensors, the cost of inertial measurement unit (IMU) steadily decline and an increasing number of robot platforms, drones and consumer electronics, especially mobile devices, are equipped with low-cost MEMS-based IMUs.

Due to the imperfections of the manufacturing and physical characteristics of the sensors, real IMU measurements are usually affected by systematic errors as well as random noise [22,23]. The process of identifying the intrinsic parameters of sensors for compensating these errors is known as IMU calibration. Although VINS have seen tremendous improvements in accuracy, robustness and efficiency

over recent years, most visual-inertial methods either treat these measurement errors as sensor noise or assume the IMU that used is precalibrated offline. For high performance inertial sensors, which generally are manufactured precisely or factory calibrated carefully with each sensor is sold with its own calibration parameters stored into the firmware, providing accurate measurements off the shelf [23], this assumption is reasonable. However, for low-cost MEMS-based inertial sensors, which are usually used for consumer mobile robots, neglecting these errors may significantly influence the performance of VINS or even breakdown the whole system. Moreover, offline IMU calibration have certain shortcomings. Traditional high precision calibration process heavily rely on special external equipment such as robot manipulator [24–26], high accuracy turntable [27], motion tracked system [28]. Such equipment is usually very expensive. As a result, the cost of one calibration process often exceeds the cost of MEMS-based IMU. Secondly, the IMU measurement models used for calibration are usually simplified linear models, the intrinsic parameters of which may vary with mechanical shocks, temperature and other factors. Treating these parameters as constant would lead to unmodeled errors and degenerate the performance of the VINS. Therefore, even high quality offline-calibration is performed, this process still needs to be repeated periodically. Moreover, performing offline calibration often needs special operators and is time consuming, which hinders the rapid deployment of VINS to consumer devices.

To this end, we propose an online IMU self-calibration method for VINS. Specifically, we are interested in concurrently performing 3D pose estimation and online IMU calibration in unknown environment, using only camera and inertial measurements, and without any knowledge about the structure of the scene or any external equipment.

We implement the method with the extension of an open source monocular VINS pipeline, VINS-Mono [8], and demonstrate that, in this setting, it is possible to concurrently localize and perform online calibration of all of the following quantities: the axis misalignments and scale factors of the IMU sensors, the g-sensitivity (also called linear acceleration dependence) of the gyroscopes. What we have to claim is that although our implementation is based on VINS-Mono, the proposed online IMU self-calibration method can be easily integrated into other graph-based visual/visual-inertial frameworks as well as the multi-camera cases. To the best of our knowledge, this is the first paper to present a VINS pipeline with online IMU self-calibration based on optimization methods. We highlight our main contributions as follows:

1. An online IMU self-calibration method for visual-inertial system that can perform in real time in unknown environment without any external equipment.
2. We frame the IMU calibration problem into general factors so that the proposed method can be easily integrated into other graph-based visual/visual-inertial frameworks.
3. An extensive evaluation on publicly available dataset showing that the online calibration can obtain accurate IMU intrinsics estimation and significantly improve the estimation precision of the VINS method compared with the estimation results with offline precalibrated IMU measurements.

The rest of this paper is organized as follows. In Section 2, we discuss the relevant literature. The algorithm as well as the implementation details are introduced in Section 3. The experimental results are shown in Section 4. Finally, the paper is concluded with the discussion and possible future research directions in Section 5.

2. Related Work

Over the last decade, there has been tremendous progress in VINS. Existing approaches can be classified into filtering-based methods [9–16] and graph-based methods [7,8,17–21]. Filtering-based methods usually process the fusion with the extended Kalman filter (EKF) and its variants, in which, the measurements from IMU are used for state propagation and observations from vision are used for update. Filtering-based methods have the advantage of fast processing since it contentiously marginalizes past states. However, their performance are usually sub-optimization due to early

fix of linearization points. Compared to filter-based methods, graph-based methods have attracted more attention in the recent years, as they obtain more accurate estimation results by maintaining measurements of long term history and perform batch optimization to obtain the optimal estimate. To achieve real time operation, graph-based methods usually apply keyframe-based techniques and optimize over a bounded-size sliding window of recent states by marginalizing past states and measurements. Nevertheless, all of these methods above either treat the IMU measurement errors as sensor noise or assume the IMU that used is precalibrated offline except the work in [13], which, to the best of our knowledge, is the only VINS method that implement IMU and camera online self-calibration in a filter-based framework. However, only simulation results were shown in this paper. Besides, their method cannot be applied for graph-based VINS methods.

For offline IMU calibration, numerous methods have been proposed over the last several decades, with the development of strapdown inertial navigation system or AHRS [22]. As we are concerned with online self-calibration, here we only give a brief review of these methods. Traditionally IMU calibration methods are usually done by using special external equipment that could provide known reference acceleration or rotational velocity of the inertial sensor. The measurements of the sensor are compared directly with the known reference value to determine the intrinsic parameters. Apparently, the accuracy of these methods strongly relies on the accuracy of the kinematic reference. To obtain highly accurate results, expensive mechanical platforms, such as robot manipulator or high accuracy turntable [24–27] are usually used, resulting in a calibration cost that often exceeds the cost of the IMU's hardware. In [28] the authors exploited using a marker-based optical tracking system to provide the reference value, while in [29], the GPS readings are used to calibrate initial biases and misalignments. In order to achieve in-situ calibration, the multi-position method was firstly introduced in [30], using the fact that the magnitude of the static acceleration must equal to the gravity's magnitude. However, this method can only calibrate the bias and scale factors of the accelerometer. This technique has been further extended by [31], in which, a method with two calibration schemes was presented. The accelerometer is firstly calibrated by exploiting the high local stability of the gravity vector's magnitude, and then the gyroscope is calibrated by comparing the gravity vector sensed by the calibrated accelerometer with the gravity vector obtained by integrating the angular velocities. The main advantage of this method is that it do not require any external mechanical equipment. A similarly approach also can be found in [23]. A self-calibration method based on an iterative matrix factorization was proposed by Hwangbo et al. [32]. this method use gravity as accelerometers reference, and a camera as gyroscopes reference. A noteworthy work in the VINS domain is the kalibr toolbox by Rehder et al. [33], in which, multiple camera-IMU extrinsic as well as the IMU intrinsics are jointly optimized in a single estimator with the continues-time batch optimization framework [33,34]. Impressive results were demonstrated in this work. However, their method cannot be integrated directly in the presented discrete-time graph-based VINS methods.

Motivated by IMU preintegration theories developed by [8,35–38], we firstly develop a novel preintegration method to handle the IMU intrinsic parameters error propagation. Then we frame the IMU preintegration measurements and IMU intrinsic parameters into general factors. In such a way, we can implement an optimization method to jointly estimate the IMU intrinsic parameters as well as the system state in a big bundle, which makes our method totally distinguished from the method in [13]. Compared to the the offline IMU calibration methods, our method integrates into the current VINS method and performs online IMU self-calibration, without any special equipment.

3. Methodology

3.1. Visual-Inertial System

Consider a sensing system (e.g., a mobile robot or a hand-held device) equipped with a monocular camera and a low-cost uncalibrated IMU. Our goal is to concurrently estimate the states of the sensing system, as well as the IMU intrinsics in a tightly-coupled graph-based framework. A general

visual-inertial system is illustrated in Figure 1, in which, several keyframes and IMU measurements are maintained in a sliding window. When a new keyframe is inserted into the sliding window, a local bundle adjustment (BA) would be implemented to jointly optimize the camera and IMU states, as well as the feature location. After optimization, one of the old keyframe as well as its corresponding states and measurements would be marginalized from the sliding windows, limiting the sliding windows size and reducing computation complexity.

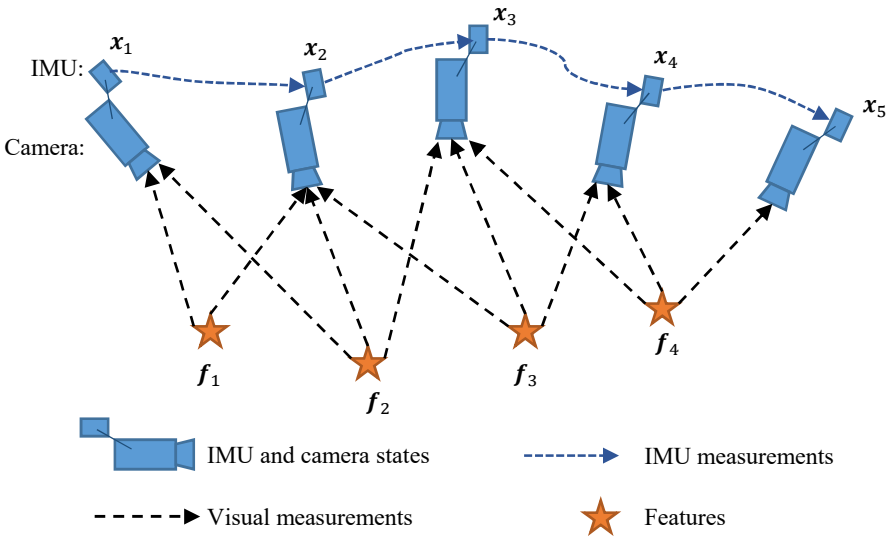

Figure 1. Illustration of visual-inertial system. Several keyframes and inertial measurement unit (IMU) measurements are maintained in a sliding window. When a new keyframe is inserted into the sliding window, a local bundle adjustment (BA) would be implemented to jointly optimize the camera and IMU states, as well as the feature location.

The notations and frame definitions used in this paper are briefly described as follows. We denote the world frame and the body frame (BF) as $(\cdot)^w$ and $(\cdot)^b$, respectively. The z-axis of the world frame is aligned with the direction of the gravity. The BF is defined to be the same as the accelerometer orthogonal frame (AOF). We primarily use quaternions $\boldsymbol{q}$ with Hamilton convention to represent rotation, but rotation matrices $\boldsymbol{R}$ are also used for the convenient representation of 3D vectors rotation. $\boldsymbol{q}_b^w$ represents the orientation of body frame with respect to the word frame. $\boldsymbol{R}_b^w$ is the corresponding rotation matrix of $\boldsymbol{q}_b^w$. The parameter $\boldsymbol{p}_b^w$ is the translation from the world frame to the body frame, being expressed in word frame, b_k is the body frame while taking the k-th image, $\boldsymbol{g}^w$ and $\boldsymbol{g}^b$ are the gravity vectors in the world frame and body frame, respectively.

The full state vector in the sliding window is defined as

$$\begin{aligned} \mathcal{X} &= [\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_n, \boldsymbol{x}_{\mathcal{I}}, \boldsymbol{f}_0, \boldsymbol{f}_1, \cdots, \boldsymbol{f}_m] \\ \boldsymbol{x}_k &= [\boldsymbol{p}_{b_k}^w, \boldsymbol{q}_{b_k}^w, \boldsymbol{v}_{b_k}^w, \boldsymbol{b}_a, \boldsymbol{b}_\omega], \end{aligned} \tag{1}$$

where $\boldsymbol{x}_k$ is the IMU state at the k-th image. It contains position, orientation and velocity of the IMU as well as the accelerometer bias $\boldsymbol{b}_a$ and gyroscope bias $\boldsymbol{b}_\omega$. The parameter $\boldsymbol{x}_{\mathcal{I}}$ is the IMU intrinsic parameters, $\boldsymbol{f}_l$ is the feature location, which can be parameterized by either the 3D position or inverse depth of the feature, n is the number of keyframes in the sliding window, m is the number of features which have been observed at least by two frames in the sliding window.

After the VINS was successfully initialized with the method proposed in [8,39], we were able to use a sliding window based framework to jointly optimize both the full system states as described in

Equation (1). We minimize the sum of prior and the Mahalanobis norm of all measurement residuals to obtain a maximum posteriori estimation as

$$\min_{\mathcal{X}} \left\{ \|\boldsymbol{r}_p - \boldsymbol{H}_p \mathcal{X}\|^2 + \sum_{k \in \mathcal{B}} \left\| \boldsymbol{r}_{\mathcal{B}} \left(\hat{\boldsymbol{z}}_{b_{k+1}}^{b_k}, \mathcal{X} \right) \right\|_{\boldsymbol{P}_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j) \in \mathcal{C}} \left\| \boldsymbol{r}_{\mathcal{C}} \left(\hat{\boldsymbol{z}}_l^{c_j}, \mathcal{X} \right) \right\|_{\boldsymbol{P}_l^{c_j}}^2 + \|\boldsymbol{r}_{\mathcal{I}}(\boldsymbol{x})\|_{\boldsymbol{P}_{\mathcal{I}}}^2 \right\}, \quad (2)$$

where $\{\boldsymbol{r}_p, \boldsymbol{H}_p\}$ are the prior information from marginalization. $\boldsymbol{r}_B(\hat{\boldsymbol{z}}_{b_{k+1}}^{b_k}, \mathcal{X})$ and $\boldsymbol{r}_C(\hat{\boldsymbol{z}}_l^{c_j}, \mathcal{X})$ are residuals for the IMU and visual measurements respectively, where $\mathcal{B}$ is the set of all IMU measurements and $\mathcal{C}$ is the set of features. The parameter $\boldsymbol{r}_{\mathcal{I}}(\boldsymbol{x})$ is the residuals for IMU intrinsic parameters.

As our goal is to estimate the IMU intrinsic parameters online, in this paper, we assume the translation and orientation between camera and IMU are fixed and known from prior calibration. Furthermore, we assume there is a vision front-end for feature detection and tracking, providing the pixel measurements of the features. Therefore, we will neglect the residuals of vision measurements $\boldsymbol{r}_C(\hat{\boldsymbol{z}}_l^{c_j}, \mathcal{X})$ in this paper and only detail the residuals of IMU measurements $\boldsymbol{r}_B(\hat{\boldsymbol{z}}_{b_{k+1}}^{b_k}, \mathcal{X})$ as well as the IMU intrinsics $\boldsymbol{r}_{\mathcal{I}}(\boldsymbol{x})$ in the following.

3.2. IMU Model

A six-degree-of-freedom IMU composes by a triple axis accelerometer and a triple axis gyroscope. Ideally, the two triple-axis define a single, shared, orthogonal 3D frame. The accelerometer senses the acceleration of the sensor along of each input axis, while gyroscope measures the angular velocity around each input axis. The scale factors convert analog or digital output of the sensor to real physical quantity measurements. Unfortunately, due to assembly imperfections, each axis of the accelerometer and gyroscope deviates by a small angle from its designed mounting direction, as a result, introducing axis misalignment and cross-axis sensitivity errors to the measurements. Also, real scale factor of each axis is unique and different from the nominal value provided by the manufacturer. In additional, the angular velocities measurements are affected by the accelerations that the sensor is subjected. Moreover, all of the measurements are almost always affected by variable bias. In order to model and compensate these errors, we employ the IMU measurement model as described in [33].

$$\boldsymbol{a}_m = \boldsymbol{S}_a \boldsymbol{M}_a (\boldsymbol{a}_{wb}^b - \boldsymbol{g}^b) + \boldsymbol{b}_a + \boldsymbol{n}_a \quad (3)$$

$$\boldsymbol{\omega}_m = \boldsymbol{S}_\omega \boldsymbol{M}_\omega \boldsymbol{\omega}_{wb}^b + \boldsymbol{A}_\omega (\boldsymbol{a}_{wb}^b - \boldsymbol{g}^b) + \boldsymbol{b}_\omega + \boldsymbol{n}_\omega, \quad (4)$$

where $\boldsymbol{a}_m$ and $\boldsymbol{\omega}_m$ are the measurement of accelerometer and gyroscope, respectively, $\boldsymbol{S}_a$ is a 3×3 diagonal matrix constructed by the scale factors of the triple axis, $\boldsymbol{M}_a$ is a 3×3 lower triangular matrix representing the axis misalignments and the cross-coupling terms, and $\boldsymbol{S}_\omega$ and $\boldsymbol{M}_\omega$ are defined analogously to $\boldsymbol{S}_a$ and $\boldsymbol{M}_a$. Each row of $\boldsymbol{M}_a$ and $\boldsymbol{M}_\omega$ is a unit vector. The parameter $\boldsymbol{A}_\omega$ is a full populated matrix representing the g-sensitivity coefficients of gyroscope, $\boldsymbol{a}_{wb}^b$ and $\boldsymbol{\omega}_{wb}^b$ are the body frame acceleration and angular velocity with respect to the word frame, being expressed in the body frame, $\boldsymbol{b}_a$ and $\boldsymbol{b}_\omega$ is the measurement bias of accelerometer and gyroscope respectively, $\boldsymbol{n}_a$, $\boldsymbol{n}_\omega$ is the measurement noise. We assume that $\boldsymbol{n}_a$, $\boldsymbol{n}_\omega$ are Gaussian white noise, $\boldsymbol{n}_a \sim \mathcal{N}(0, \sigma_a^2)$, $\boldsymbol{n}_\omega \sim \mathcal{N}(0, \sigma_\omega^2)$. For the derivations of Equations (3) and (4), please refer to [33].

The accelerometer and gyroscope bias are modeled as random walks, the derivatives of which are Gaussian white noise, $\boldsymbol{n}_{b_a} \sim \mathcal{N}(0, \sigma_{b_a}^2)$, $\boldsymbol{n}_{b_\omega} \sim \mathcal{N}(0, \sigma_{b_\omega}^2)$.

$$\dot{\boldsymbol{b}}_a = \boldsymbol{n}_{b_a}, \quad \dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b_\omega}. \quad (5)$$

It is inconvenient to optimize the $\boldsymbol{M}_a$ and $\boldsymbol{M}_\omega$ as each row of these two matrices is constrained to a unit vector. Besides, when integrating the angular velocity and acceleration to obtain the motion of the system, we need to compute the inverse matrix of $\boldsymbol{S}_a$, $\boldsymbol{M}_a$, $\boldsymbol{S}_\omega$, $\boldsymbol{M}_\omega$ at every step to compensate the measurement

errors, which is a waste of computing resources. If defining $\boldsymbol{T}_a = (\boldsymbol{S}_a\boldsymbol{M}_a)^{-1}$, $\boldsymbol{T}_\omega = (\boldsymbol{S}_\omega\boldsymbol{M}_\omega)^{-1}$, we could get the compensated acceleration and angular velocity from the measurements more effectively:

$$\boldsymbol{a}_{wb}^b = \boldsymbol{T}_a(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) + \boldsymbol{g}^b \tag{6}$$

$$\boldsymbol{\omega}_{wb}^b = \boldsymbol{T}_\omega\big(\boldsymbol{\omega}_m - \boldsymbol{A}_\omega(\boldsymbol{a}_{wb}^b - \boldsymbol{g}^b) - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega\big), \tag{7}$$

where, $\boldsymbol{T}_a$ is still a lower triangular matrix, $\boldsymbol{T}_\omega$ is a full populated matrix without constraint. For an uncalibrated IMU, $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$ is roughly close to identify matrix with some uncertainty. $\boldsymbol{A}_\omega$ is close to $\mathbf{0}$.

3.3. IMU Preintegration

The IMU preintegration method was firstly introduced in [35], which parameterize the rotation error using Euler angles. This method was further improved in [8] with the continuous-time quaternion-based derivation and including the handling of IMU biases. The intuitive concept of IMU preintegration is illustrated in Figure 2. The proposed an IMU preintegration method that is motivated by [8], but is different from the previous work as we introduce the IMU intrinsic parameters in the measurement model. Therefore, we need to re-derive all of the equations from scratch.

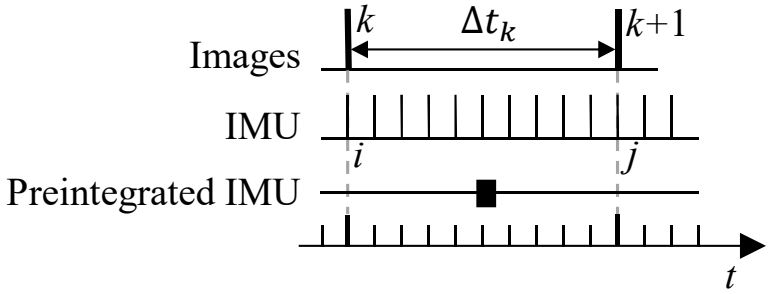

Figure 2. Illustration of the concept of IMU preintegration. The IMU measures the acceleration and angular velocity at discrete time. Generally, The IMU measurement frequency is much larger than the frame rate of camera. Giving two time consecutive frames b_k and b_{k+1}, there exists several measurements in time interval $[t_k, t_{k+1}]$. The IMU measurements between image frames k and $k+1$ are pre-integrated into a single compound measurement. From an optimization perspective, it can be seen as the observations or measurements with which we can construct the error terms and adjust the corresponding state values to minimize the errors.

Theoretically, giving two time instants at images frames k and $k+1$, we can compute the states at t_{k+1} as a function of IMU measurements between two frames:

$$\boldsymbol{p}_{b_{k+1}}^w = \boldsymbol{p}_{b_k}^w + \boldsymbol{v}_{b_k}^w\Delta t_k + \frac{1}{2}\boldsymbol{g}^w\Delta t_k^2 + \iint_{t\in[k,k+1]} R_{b_t}^w\boldsymbol{T}_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t})\,\mathrm{d}t^2 \tag{8}$$

$$\boldsymbol{v}_{b_{k+1}}^w = \boldsymbol{v}_{b_k}^w + \boldsymbol{g}^w\Delta t_k + \int_{t\in[k,k+1]} \boldsymbol{R}_{b_t}^w\boldsymbol{T}_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t})\,\mathrm{d}t \tag{9}$$

$$\boldsymbol{q}_{b_{k+1}}^w = \boldsymbol{q}_{b_k}^w \otimes \int_{t\in[k,k+1]} \frac{1}{2}\boldsymbol{q}_{b_t}^w \otimes \boldsymbol{T}_\omega\big(\boldsymbol{\omega}_{m_t} - \boldsymbol{A}_\omega T_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t}) - \boldsymbol{b}_{\omega_t} - \boldsymbol{n}_{\omega_t}\big)\,\mathrm{d}t, \tag{10}$$

where Δt_k is the duration of time interval $[t_k, t_{k+1}]$.

While Equations (8)–(10) provide an estimation of motion between time t_k and t_{k+1}, it can be seen that the IMU state propagation requires the rotation, position and velocity of frame k as the starting states. In the optimization-based algorithm, we adjust the full system states at every iteration step, which means these starting states may change. In such cases, we need to re-integrate the IMU measurements, which is computation consuming. To avoid this, we adopt the preintegration method.

Premultiply $\boldsymbol{q}_w^{b_k}$ or $\boldsymbol{R}_w^{b_k}$ to both sides of Equations (8)–(10) to change the reference frame for the IMU propagation to the local frame b_k:

$$\boldsymbol{R}_w^{b_k}\boldsymbol{p}_{b_{k+1}}^w = \boldsymbol{R}_{b_k}^w(\boldsymbol{p}_{b_k}^w + \boldsymbol{v}_{b_k}^w\Delta t + \frac{1}{2}\boldsymbol{g}^w\Delta t^2) + \boldsymbol{\alpha}_{b_{k+1}}^{b_k} \tag{11}$$

$$\boldsymbol{R}_w^{b_k}\boldsymbol{v}_{b_{k+1}}^w = \boldsymbol{R}_w^{b_k}\boldsymbol{v}_{b_k}^w + \boldsymbol{R}_w^{b_k}\boldsymbol{g}^w\Delta t_k + \boldsymbol{\beta}_{b_{k+1}}^{b_k} \tag{12}$$

$$\boldsymbol{q}_w^{b_k} \otimes \boldsymbol{q}_{b_{k+1}}^w = \boldsymbol{\gamma}_{b_{k+1}}^{b_k}, \tag{13}$$

where

$$\boldsymbol{\alpha}_{b_{k+1}}^{b_k} = \iint_{t\in[k,k+1]} \boldsymbol{R}_{b_t}^{b_k}\boldsymbol{T}_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t})\,\mathrm{d}t^2 \tag{14}$$

$$\boldsymbol{\beta}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \boldsymbol{R}_{b_t}^{b_k}\boldsymbol{T}_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t})\,\mathrm{d}t \tag{15}$$

$$\boldsymbol{\gamma}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \frac{1}{2}\boldsymbol{\gamma}_{b_t}^{b_k} \otimes \boldsymbol{T}_\omega\left(\boldsymbol{\omega}_{m_t} - \boldsymbol{A}_\omega\boldsymbol{T}_a(\boldsymbol{a}_{m_t} - \boldsymbol{b}_{a_t} - \boldsymbol{n}_{a_t}) - \boldsymbol{b}_{\omega_t} - \boldsymbol{n}_{\omega_t}\right)\mathrm{d}t. \tag{16}$$

The preintegration parts $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$, $\boldsymbol{\beta}_{b_{k+1}}^{b_k}$, $\boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ can be obtained solely with IMU measurements and its intrinsic parameters. These preintegrated measurements construct the constraints between the states at time t_k and the states at time t_{k+1}. Therefore, we can construct the error terms with these preintegrated IMU measurements, and adjust the corresponding state values to minimize the errors base on optimization method.

It can be seen that the preintegration measurements in Equations (14)–(16) are also the function of the IMU intrinsic parameters, $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ as well as the bias $\boldsymbol{b}_a$, $\boldsymbol{b}_\omega$. As a result, the imperfection of these parameters would introduce errors into the measurements. This also means that we can concurrently optimize the IMU intrinsic parameters to minimize the errors constrained by the preintegration measurements. To achieve this goal, two problems need to be dealt with:

1. Compute the Jacobian matrix of $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$, $\boldsymbol{\beta}_{b_{k+1}}^{b_k}$, $\boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ with respect to the IMU intrinsic parameters. The Jacobian matrix will be used in two ways: (1) for the optimization method, the Jacobian matrix would be used to evaluate the increment of the input arguments; (2) at the end of each iteration step, we use Jacobian as well as the new adjusted states will be used to compute the first-order approximation of preintegration measurement so that we can avoid re-integrating the IMU measurements.
2. Estimate the uncertainty of the preintegration measurements. All of the IMU measurements contain random noises. These noises, as well as the the inaccuracy of the IMU intrinsic parameters, introduce the uncertainty to the preintegration results. In order to get more precise optimization results, we need to estimate the uncertainty of the preintegration measurements in the form of a covariance matrix or information matrix and use these uncertainty information to weight the error terms during optimization.

These two problems will be tackled in the next section.

3.4. Jacobian and Noise Propagation

In order to compute the Jacobian matrix and estimate the uncertainty of preintegration results, we divide the true state value of preintegration result $\boldsymbol{x}$ into two components, the nominal state $\hat{\boldsymbol{x}}$ and the error state $\delta\boldsymbol{x}$. The nominal state is estimated by integrating the IMU measurements directly, without taking into account the noise terms and imperfect intrinsic parameters or unmodeled error. As a results, it will accumulate errors. These errors are collected into the error state. In this way, the true state can be expressed as a composition of the nominal state and error state, that is,

$$\boldsymbol{x} = \hat{\boldsymbol{x}} \oplus \delta\boldsymbol{x}, \tag{17}$$

where $\oplus$ is the composition operator. Since the quaternion $\boldsymbol{q}$ is over-parameterized, we define its error term with the minimal representation $\delta\boldsymbol{\theta}$. As a result, for quaternion, $\oplus$ is defined as

$$\boldsymbol{q} = \hat{\boldsymbol{q}} \oplus \delta\boldsymbol{\theta} \approx \hat{\boldsymbol{q}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} \Rightarrow \delta\boldsymbol{\theta} = 2[\hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}]_{xyz}, \tag{18}$$

where $[\cdot]_{xyz}$ extracts the vector part of a quaternion $\boldsymbol{q}$. For other states, $\hat{\boldsymbol{x}} \oplus \delta\hat{\boldsymbol{x}} \triangleq \hat{\boldsymbol{x}} + \delta\boldsymbol{x}$.

The concept of nominal-state and error-state is similar to the error-state Kalman filter (ESKF) [40,41]. The main difference is that we need to tackle the imperfection of the IMU intrinsic parameters. This is a nontrivial extension as it introduces 24 more parameters into the state space, which makes the equations more complex compared to the general ESKF and preintegration methods showed in [8,37,38].

The nominal-state is computed as follows:

$$\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} = \iint_{t\in[k,k+1]} \hat{\boldsymbol{R}}_{b_t}^{b_k} \hat{\boldsymbol{T}}_a (\boldsymbol{a}_{m_t} - \hat{\boldsymbol{b}}_{a_t}) \, \mathrm{d}t^2 \tag{19}$$

$$\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \hat{\boldsymbol{R}}_{b_t}^{b_k} \hat{\boldsymbol{T}}_a (\boldsymbol{a}_{m_t} - \hat{\boldsymbol{b}}_{a_t}) \, \mathrm{d}t \tag{20}$$

$$\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} = \int_{t\in[k,k+1]} \frac{1}{2} \hat{\boldsymbol{\gamma}}_{b_t}^{b_k} \otimes \hat{\boldsymbol{T}}_\omega \left(\boldsymbol{\omega}_{m_t} - \hat{\boldsymbol{A}}_\omega \hat{T}_a (\boldsymbol{a}_{m_t} - \hat{\boldsymbol{b}}_{a_t}) - \hat{\boldsymbol{b}}_{\omega_t}\right) \mathrm{d}t. \tag{21}$$

As the real IMU measures the acceleration and angular velocity at discrete time, Equations (19)–(21) can be computed by numberical integraion methods such as Euler, mid-point or Runge—Kutta methods. The mid-point integration is used in our implementation code, which is detailed in the Appendix A.2. At the beginning of integration, $\boldsymbol{\alpha}_{b_k}^{b_k}$, $\boldsymbol{\beta}_{b_k}^{b_k}$ is set to $\mathbf{0}$, and $\boldsymbol{\gamma}_{b_k}^{b_k}$ is set to identity quaternion.

With the computation above, all the large-signal dynamics have been integrated in the nominal-state. As a result, the error-state is always small. We can solve its kinematics and neglect all second-order infinitesimals to get the continuous-time linearized dynamics.

$$\delta\dot{\boldsymbol{\alpha}}_{b_t}^{b_k} = \delta\boldsymbol{\beta}_{b_t}^{b_k} \tag{22}$$

$$\delta\dot{\boldsymbol{\beta}}_{b_t}^{b_k} = -\hat{\boldsymbol{R}}_{b_t}^{b_k} [\hat{T}_a \hat{\boldsymbol{a}}]_\times \delta\theta_{b_t}^{b_k} - \hat{\boldsymbol{R}}_{b_t}^{b_k} \hat{\boldsymbol{T}}_a \delta\hat{\boldsymbol{b}}_a + \hat{\boldsymbol{R}}_{b_t}^{b_k} \delta\boldsymbol{T}_a \hat{\boldsymbol{a}}_B - \hat{\boldsymbol{R}}_{b_t}^{b_k} \boldsymbol{T}_a \boldsymbol{n}_a \tag{23}$$

$$\begin{aligned}\delta\dot{\boldsymbol{\theta}}_{b_t}^{b_k} = &-[\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \delta\boldsymbol{\theta}_{b_t}^{b_k} + \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \hat{\boldsymbol{T}}_a \delta\boldsymbol{b}_a - \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{b}_\omega - \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \delta\boldsymbol{T}_a \hat{\boldsymbol{a}}_B \\ &+ \delta\boldsymbol{T}_\omega \hat{\boldsymbol{\omega}}_B - \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{A}_\omega \boldsymbol{T}_a \hat{\boldsymbol{a}}_B + \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \hat{\boldsymbol{T}}_a \boldsymbol{n}_a - \hat{\boldsymbol{T}}_\omega \boldsymbol{n}_\omega,\end{aligned} \tag{24}$$

where $\hat{\boldsymbol{R}}_{b_t}^{b_k}$ is the corresponding rotation matrix of $\boldsymbol{\gamma}_{b_t}^{b_k}$, $\hat{\boldsymbol{a}}_B = (\boldsymbol{a}_{m_t} - \hat{\boldsymbol{b}}_{a_t})$, $\hat{\boldsymbol{\omega}}_B = \boldsymbol{\omega}_{m_t} - \hat{\boldsymbol{A}}_\omega \hat{T}_a (\boldsymbol{a}_{m_t} - \hat{\boldsymbol{b}}_{a_t}) - \hat{\boldsymbol{b}}_{\omega_t}$, $[\cdot]_\times$ is the skew symmetry matrix operator,

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_x & \omega_y \\ \omega_x & 0 & -\omega_z \\ -\omega_y & \omega_z & 0 \end{bmatrix}. \tag{25}$$

Details about how to get the error-state kinematic representation above is given in the Appendix A.1.

The error-state dynamics of the IMU intrinsic parameters are as follows

$$\delta\dot{\boldsymbol{b}}_a = \boldsymbol{n}_{b_a}, \quad \delta\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b_\omega} \tag{26}$$

$$\delta\dot{\boldsymbol{T}}_a = \mathbf{0}, \quad \delta\dot{\boldsymbol{T}}_\omega = \mathbf{0}, \quad \delta\dot{\boldsymbol{A}}_\omega = \mathbf{0}. \tag{27}$$

In order to compute the the Jacobian matrix and covariance recursively, we need to convert Equations (22)–(24), (26) and (27) to matrix representation. This takes a little trick.

Define

$$\mathcal{X}_{pre} = [\boldsymbol{\alpha}_{b_t}^{b_k}, \boldsymbol{\beta}_{b_t}^{b_k}, \boldsymbol{\gamma}_{b_t}^{b_k}, \boldsymbol{b}_a, \boldsymbol{b}_\omega, \boldsymbol{t}_a, \boldsymbol{t}_\omega, \boldsymbol{a}_\omega], \tag{28}$$

where

$$\begin{aligned}
\boldsymbol{t}_a &= [T_{a_{11}}, T_{a_{21}}, T_{a_{22}}, T_{a_{31}}, T_{a_{32}}, T_{a_{33}}]^T \\
\boldsymbol{t}_\omega &= [T_{\omega_{11}}, T_{\omega_{12}}, T_{\omega_{13}}, T_{\omega_{21}}, T_{\omega_{22}}, T_{\omega_{23}}, T_{\omega_{31}}, T_{\omega_{32}}, T_{\omega_{33}}]^T \\
\boldsymbol{a}_\omega &= [A_{\omega_{11}}, A_{\omega_{12}}, A_{\omega_{13}}, A_{\omega_{21}}, A_{\omega_{22}}, A_{\omega_{23}}, A_{\omega_{31}}, A_{\omega_{32}}, A_{\omega_{33}}]^T.
\end{aligned}$$

With $T_{a_{ij}}$, $T_{\omega_{ij}}$, $A_{\omega_{ij}}$ are the corresponding elements of the matrix $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$, respectively. Assume $\boldsymbol{r}_{3\times1} = [r_1, r_2, r_3]^T$. Now we define the $[\cdot]_T$ operator as

$$[\boldsymbol{r}_{3\times1}]_T = \begin{bmatrix} r_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_1 & r_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_1 & r_2 & r_3 \end{bmatrix}_{3\times6}, \text{ such that } \boldsymbol{T}_a\boldsymbol{r}_{3\times1} = [\boldsymbol{r}_{3\times1}]_T\boldsymbol{t}_a$$

$$\text{and } [\boldsymbol{r}_{3\times1}]_T = \begin{bmatrix} \boldsymbol{r}^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{r}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{r}^T \end{bmatrix}_{3\times9}, \text{ such that } \boldsymbol{T}_\omega\boldsymbol{r}_{\boldsymbol{3\times1}} = [\boldsymbol{r}_{\boldsymbol{3\times1}}]_T\boldsymbol{t}_\omega, \boldsymbol{A}_\omega\boldsymbol{r}_{\boldsymbol{3\times1}} = [\boldsymbol{r}_{\boldsymbol{3\times1}}]_T\boldsymbol{a}_\omega.$$

With the definitions above, Equations (22)–(24), (26) and (27) can be converted to matrix representation form

$$\delta\dot{\mathcal{X}}_{pre} = \boldsymbol{F}_t\delta\mathcal{X}_{pre} + \boldsymbol{G}_t\boldsymbol{n} \tag{29}$$

where

$$\boldsymbol{F}_t = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\hat{\boldsymbol{R}}_{b_t}^{b_k}[\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}]_\times & -\hat{\boldsymbol{R}}_{b_t}^{b_k}\hat{\boldsymbol{T}}_a & \boldsymbol{0} & \hat{\boldsymbol{R}}_{b_t}^{b_k}[\hat{\boldsymbol{a}}_t]_T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & [\hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{\omega}}_t]_\times & \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a & \hat{\boldsymbol{T}}_\omega & -\hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega[\hat{\boldsymbol{a}}_t]_T & [\hat{\boldsymbol{\omega}}]_T & \hat{\boldsymbol{T}}_\omega[\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_t]_T \\ \multicolumn{8}{c}{\boldsymbol{0}_{30\times39}} \end{bmatrix}$$

$$\boldsymbol{G}_t = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a & -\hat{\boldsymbol{R}}_{b_t}^{b_k}\hat{\boldsymbol{T}}_a & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\hat{\boldsymbol{T}}_\omega & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \multicolumn{4}{c}{\boldsymbol{0}_{24\times12}} \end{bmatrix},$$

and $\boldsymbol{n} = [\boldsymbol{n}_a, \boldsymbol{n}_\omega, \boldsymbol{n}_{b_a}, \boldsymbol{n}_{b_\omega}]$.

With the Equation (29), it is convenient to compute the the covariance matrix $\boldsymbol{P}_{b_{k+1}}^{b_k}$ recursively by the first order discrete-time covariance update

$$\boldsymbol{P}_{b_{t+\delta t}}^{b_k} = (\boldsymbol{I} + \boldsymbol{F}_t\delta t)\boldsymbol{P}_{b_t}^{b_k}(\boldsymbol{I} + \boldsymbol{F}_t\delta t)^T + (\boldsymbol{G}_t\delta t)\boldsymbol{Q}(\boldsymbol{G}_t\delta t)^T, \tag{30}$$

where δt is the time between two IMU measurements, and $\boldsymbol{Q}$ is the diagonal covariance matrix of noise $(\sigma_a^2, \sigma_g^2, \sigma_{b_a}^2, \sigma_{b_g}^2)$. The initial covariance value corresponding to $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{b}_a$, $\boldsymbol{b}_\omega$ is set to 0. For $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$, the initial covariance can be set to 0 or a constant uncertainty, which will be discussed in Section 3.6.

Now we deal with the Jacobian matrix. Neglecting the noise term, Equation (29) can be converted to a discrete-time representation as following:

$$\delta\mathcal{X}_{pre}(t+\delta t) = (1 + \boldsymbol{F}_t\delta t)\delta\mathcal{X}_{pre}(t). \tag{31}$$

As a result, the Jacobian of $\delta\mathcal{X}_{pre}(t+\delta t)$ with respect to $\delta\mathcal{X}_{pre}(t)$ can be obtained conveniently:

$$\boldsymbol{J}_t^{t+\delta t} = \frac{\partial\delta\partial\mathcal{X}_{pre}(t+\delta t)}{\delta\mathcal{X}_{pre}(t)} = 1 + \boldsymbol{F}_t\delta t. \tag{32}$$

We can compute J_k^{k+1} recursively:

$$\begin{aligned}\boldsymbol{J}_k^{k+1} &= \frac{\partial\delta\mathcal{X}_{pre}(j)}{\partial\delta\mathcal{X}_{pre}(j-1)}\frac{\partial\delta\mathcal{X}_{pre}(j-1)}{\partial\delta\mathcal{X}_{pre}(j-2)}\cdots\frac{\partial\delta\mathcal{X}_{pre}(i+1)}{\partial\delta\mathcal{X}_{pre}(i)} \\ &= (I + \boldsymbol{F}_{t,j-1}\delta t_{j-1})(I + \boldsymbol{F}_{t,j-2}\delta t_{j-2})\cdots(I + \boldsymbol{F}_i\delta t_i),\end{aligned} \tag{33}$$

where i and j represent the i-th and j-th IMU measurement obtained at the time instant of frame k and $k+1$, respectively, as illustrated in Figure 2.

The first order approximation of $\boldsymbol{\alpha}_{b_{k+1}}$, $\boldsymbol{\beta}_{b_{k+1}}$, $\gamma_{b_{k+1}}$ with respect to the IMU intrinsic parameters can be computed as:

$$\boldsymbol{\alpha}_{b_{k+1}}^{b_k} = \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_a}^{\alpha}\delta\boldsymbol{b}_{a_k} + \boldsymbol{J}_{b_\omega}^{\alpha}\delta\boldsymbol{b}_{\omega_k} + \boldsymbol{J}_{t_a}^{\alpha}\delta\boldsymbol{t}_{a_k} + \boldsymbol{J}_{t_\omega}^{\alpha}\delta\boldsymbol{t}_{\omega_k} + \boldsymbol{J}_{a_\omega}^{\alpha}\delta\boldsymbol{a}_{\omega_k} \tag{34}$$

$$\boldsymbol{\beta}_{b_{k+1}}^{b_k} = \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \boldsymbol{J}_{b_a}^{\beta}\delta\boldsymbol{b}_{a_k} + \boldsymbol{J}_{b_\omega}^{\beta}\delta\boldsymbol{b}_{\omega_k} + \boldsymbol{J}_{t_a}^{\beta}\delta\boldsymbol{t}_{a_k} + \boldsymbol{J}_{t_\omega}^{\beta}\delta\boldsymbol{t}_{\omega_k} + \boldsymbol{J}_{a_\omega}^{\beta}\delta\boldsymbol{a}_{\omega_k} \tag{35}$$

$$\gamma_{b_{k+1}}^{b_k} = \hat{\gamma}_{b_{k+1}}^{b_k} \otimes (\boldsymbol{J}_{b_a}^{\gamma}\delta\boldsymbol{b}_{a_k} + \boldsymbol{J}_{b_\omega}^{\gamma}\delta\boldsymbol{b}_{\omega_k} + \boldsymbol{J}_{t_a}^{\gamma}\delta\boldsymbol{t}_{a_k} + \boldsymbol{J}_{t_\omega}^{\gamma}\delta\boldsymbol{t}_{\omega_k} + \boldsymbol{J}_{a_\omega}^{\gamma}\delta\boldsymbol{a}_{\omega_k}), \tag{36}$$

where $\boldsymbol{J}_{b_a}^{\alpha}$ is the sub-block matrix of $\boldsymbol{J}_{b_{k+1}}$ whose location is corresponding to $\frac{\partial\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}}{\boldsymbol{b}_{a_k}}$. The same meaning is also used for other $\boldsymbol{J}$ in the above.

At each step of optimization, we can use Equations (34)–(36) to update the value of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, γ with the new estimated IMU intrinsic parameters and recompute the corresponding residuals.

3.5. IMU Measurement Factor

Considering two consecutive frames k and $k+1$ in the window, the residual of preintegrated IMU measurement can be constructed from Equations (11)–(13):

$$\boldsymbol{r}_B(\hat{\boldsymbol{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) = \begin{bmatrix}\delta\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \delta\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \delta\hat{\gamma}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{b}_a \\ \delta\boldsymbol{b}_\omega\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}_w^{b_k}(\boldsymbol{p}_{b_{k+1}}^w - \boldsymbol{p}_{b_k}^w + \frac{1}{2}\boldsymbol{g}^w\Delta t^2 - \boldsymbol{v}_{b_k}^w\Delta t) - \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \boldsymbol{R}_w^{b_k}(\boldsymbol{v}_{b_{k+1}}^w + \boldsymbol{g}^w\Delta t - \boldsymbol{v}_{b_k}^w) - \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ 2[(\hat{\gamma}_{b_{k+1}}^{b_k})^{-1} \otimes (\boldsymbol{q}_{b_k}^w)^{-1} \otimes \boldsymbol{q}_{b_{k+1}}^w]_{xyz} \\ \boldsymbol{b}_{a,k+1} - \boldsymbol{b}_{a,k} \\ \boldsymbol{b}_{\omega,k+1} - \boldsymbol{b}_{\omega,k}\end{bmatrix}. \tag{37}$$

Please note that the accelerometer and gyroscope bias are also included in the residual terms for online correction. Equation (37) seems the same as the IMU measurements residual in the paper [8]. However, as our preintegrated IMU measurements $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, γ handle the imperfection of the IMU intrinsic parameters, which made it different from the one of [8]. Besides, the Jacobian of $\boldsymbol{r}_B(\hat{\boldsymbol{z}}_{b_{k+1}}^{b_k}, \mathcal{X})$ with respect to the input arguments is completely different, too.

3.6. IMU Intrinsic Factor

There are several possible but different strategies for defining the IMU intrinsic parameters residual:

1. The parameters $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ are modeled as random-walk processes, but are constant between two frames in the sliding window, and are handled with the same strategy as the IMU bias.
2. The parameters $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ are assumed to be uncertain but constant between two frames in the sliding window. With this strategy, we can set the covariance value of $\boldsymbol{P}_{b_k}^{b_k}$ in Equation (30) corresponding to $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ with a constant value and optimize them at every frame.
3. The parameters $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ are assumed to be uncertain, but constant in the timespan of the sliding window.

From the perspective of optimization method, the strategies 1 and 2 are nearly same, except that the covariance value is different, with strategy 1 computing the covariance during preintegration. However, for every frame these two strategies would add additional 24 parameters to the estimator. Assuming the size of sliding window is 10, this means that we will need to optimize an additional 240 parameters at each step, leading to a computational problem for realtime implementation. Besides, compared to the accelerometer and gyroscope bias which vary with time and temperature, the scale factor and misalignment parameters seem more stable, which means that we do not need to optimize them at every frame. Moreover, the scale factors and axis misalignments may couple with the measurement bias under degenerate motions such as rectilinear trajectory, motion with zero acceleration or rotation. In such a case, it is impossible to distinguish all these parameters in one or two frames. With these reasons, in our algorithm, we would choose the strategy 3.

The residual of $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ in our implementation is defined as follows:

$$\boldsymbol{r}_{\mathcal{I}}(\mathcal{X}) = \begin{bmatrix} \boldsymbol{t}_a - \hat{\boldsymbol{t}}_{a,k} \\ \boldsymbol{t}_\omega - \hat{\boldsymbol{t}}_{\omega,k} \\ \boldsymbol{a}_\omega - \hat{\boldsymbol{a}}_{\omega,k} \end{bmatrix}, \tag{38}$$

where $\hat{\boldsymbol{t}}_{a,k}$, $\hat{\boldsymbol{t}}_{\omega,k}$ and $\hat{\boldsymbol{a}}_{\omega,k}$ are the current estimation of IMU intrinsic parameters. Please note that the residual is weighted by a constant covariance matrix $\boldsymbol{P}_{\mathcal{I}}$ and add to the cost function, Equation (2).

3.7. Discussion and Implementation Details

The considered IMU intrinsic parameters in our method include axis misalignments, scale factors and g-sensitivity. In practice, however, it should be noted that online g-sensitivity calibration is a difficult task. This my be caused by various reasons. First, the errors caused by g-sensitivity are generally smaller than other errors, especially in cases that the IMU suffers from motion with low acceleration or constant speed, or remains static. Second, the g-sensitivity coefficients of a real IMU may vary with the frequency of vibration. As a result, the errors caused by g-sensitivity are more prone to couple with the gyroscope bias or manifest as random noise. In some worse cases, online g-sensitivity calibration may lead to less accuracy of the motion estimation of VINS. Moreover, g-sensitivity calibration would introduce an extra nine variables to the state space, which would consume more computational resource. Therefore, in practice application, whether or not calibrating the g-sensitivity online depends on the IMU used and the motion that the sensor system are subjected. Fortunately, it is very easy to calibrate without g-sensitivity by setting the g-sensitivity coefficients to zeros and keep them constant. The implementation details would be discussed in the following.

We implemented the proposed online IMU self-calibration method with the extension of VINS-Mono [8], as depicted in Figure 3. The system started with measurement preprocessing. For each new image, features were detected by the Shi–Tomasi corner detector [42] and tracked by a KLT tracker [43]. Meanwhile, IMU measurements were locally pre-integrated. The initialization procedure provided all necessary values for bootstrapping the subsequent nonlinear optimization. After the system has been successfully initialized, the tightly-coupled nonlinear optimization would be implemented when every frame is added to the sliding window. After that, keyframe management module marginalized one of the frames from the sliding window in order to bound the computation complexity. We add the IMU intrinsic parameters into the state vector and replace the

IMU preintegration block with our method, which are detailed in Sections 3.3 and 3.4. The proposed IMU measurement factor (Section 3.5) and IMU intrinsic factor (Section 3.6) were used in the nonlinear optimization block. The optimization computation is based on the Ceres Solver [44]. In the implementation, we separate the $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ into individual parameter blocks so that we can conveniently set any blocks to constant when optimization, depending the real application. In such way, we can implement our method with and without g-sensitivity calibration, as discussed above.

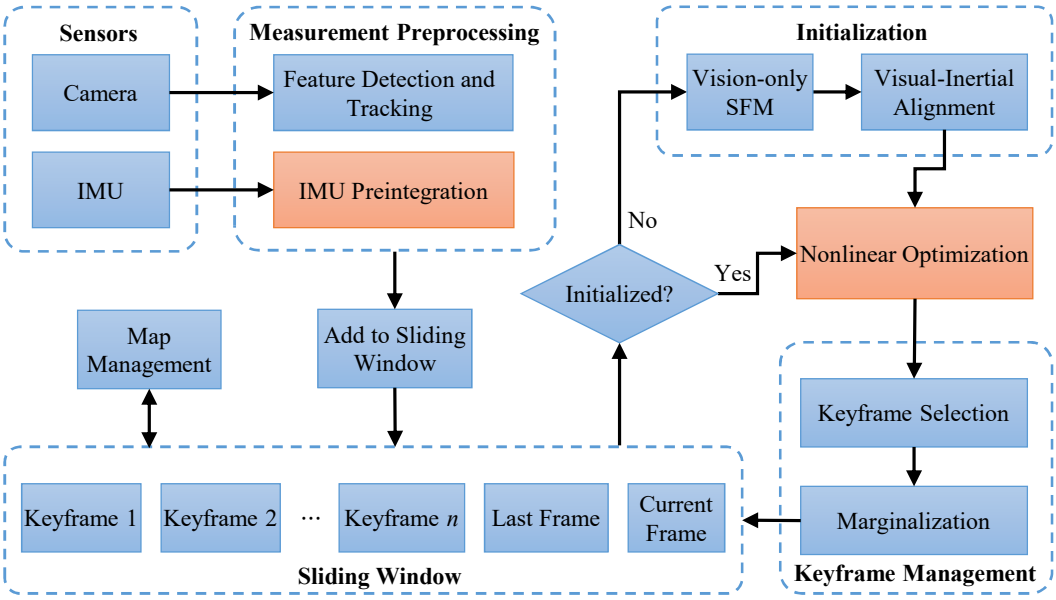

Figure 3. Block diagram illustrating of a monocular visual-inertial system (VINS-Mono) [8], based on which, we implement the proposed online IMU self-calibration method. We add the IMU intrinsic parameters into the state vector and replace the IMU Preintegration block with our method, which are detailed in Sections 3.3 and 3.4. The proposed IMU measurement factor (Section 3.5) and IMU intrinsic factor (Section 3.6) are used in the nonlinear optimization block.

4. Experimental Results

We performed dataset experiments to evaluate the proposed online IMU self-calibration method. At present there are several publicly available datasets for VINS evaluation, such as UMich NCLT [45], EuRoc [46], PennCOSYVIO [47], Zurich Urban MAV [48], TUM VI [49], etc., in which, EuRoc is a very popular dataset and used by many papers [8,20,38,39,50–52]. For a completely comparison of these datasets, please refer to [49]. In this paper we adopt the TUM VI dataset [49] for two main reasons: (1) the IMU used in EuRoc dataset is an ADIS16448, a high-end industrial grade IMU with precisely factory calibrated. In contrast, the TUM VI uses a low-cost consumer grade IMU, Bosch BMI160, which makes it more suitable for evaluating our algorithm; (2) The dataset provides both the raw IMU measurements and the precalibrated IMU measurements that compensated for proper IMU scaling and axis alignment. The value of scale factor and axis alignment parameters used for compensation are also provided with the dataset. As a result, we can easily run our algorithm with the raw IMU measurements and compare the estimated intrinsic parameters with the provided reference value.

The TUM VI dataset provides several categories of sequences for the evaluation of VINS method, including three corridor sequences, six magistrale sequences, eight outdoor sequences, six room sequences, and three slider sequence. The outdoor sequences are neglected in our experiments as we focus on the IMU self-calibration but not long time drift evaluation in challenging environments. To perform the comparison experiments, we firstly packed the the quarter resolution images (512 × 512 pixels) of the left camera, precalibred IMU measurements as well as the raw IMU measurements with different topic names into one single ROS bag file for each sequence so that we can easily evaluate the algorithm with different type of IMU measurements by just modifying the topic name.

Please note that we deduct the large IMU bias, which is coarsely reproducible between sensor restarts, from the raw measurements. This is reasonably explained in [49]. We also compensate the timestamp of IMU data so that the timestamp is consistent for camera, IMU and ground truth pose.

4.1. IMU Intrinsics Estimation Results

As discussed in Section 3.7, it is difficult to calibrate the g-sensitivity online. In the experiments, we ran the algorithm two times on each sequence of dataset, with and without $\boldsymbol{A}_\omega$ estimation, respectively. Then we compared the IMU intrinsics estimation results with the the reference values. In all experiments, the initial value of $\boldsymbol{T}_a$ and $\boldsymbol{T}_\omega$ were set to the identity matrix, with the covariance as 2.0×10^{-4}, and the initial value $\boldsymbol{A}_\omega$ was set as a zeros matrix with covariance as 5.0×10^{-6}.

4.1.1. IMU Intrinsics Estimation without G-Sensivity

Figures 4 and 5 show the estimation results on the first sequence of each categories, corridor 1, magistrale 1, room 1, slides 1, respectively, in detail. We can see that all parameters converge to a certain range close to the ref value, but slightly vary with time after that. For each sequence, the convergence speed is different. Generally, the parameters of the gyroscope converge in about 15–20 s and the estimation results is more stable. In contrast, it takes 50–100 s for the parameters of accelerometer to converge. In Figures 4b and 5b, the estimation results of $T_{a_{21}}$ and $T_{a_{32}}$ show larger differences compared with the results of Figures 4a and 5a.

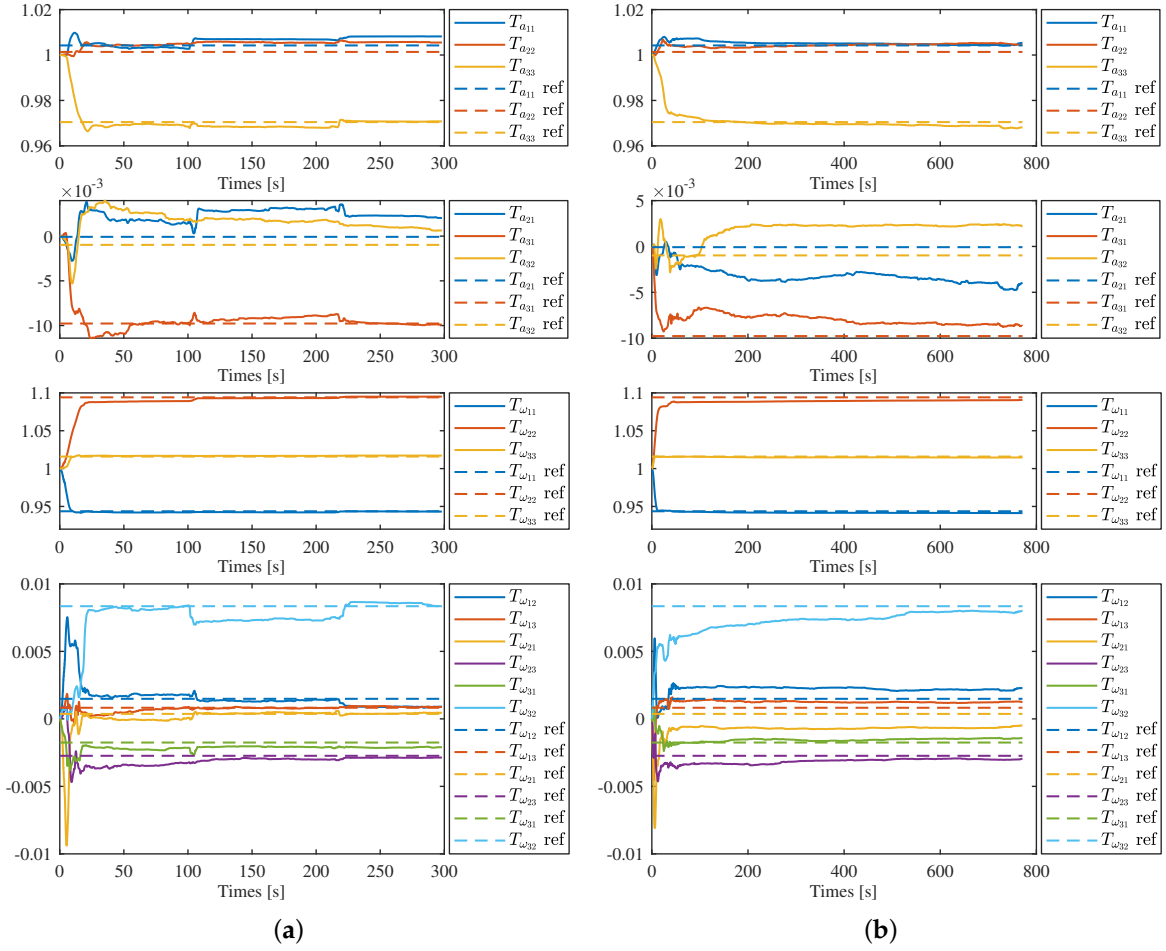

Figure 4. IMU intrinsics estimation in corridor 1 (**a**) and magistrale 1 (**b**). Solid lines are the estimation results and the dotted lines with the same color are the corresponding reference value.

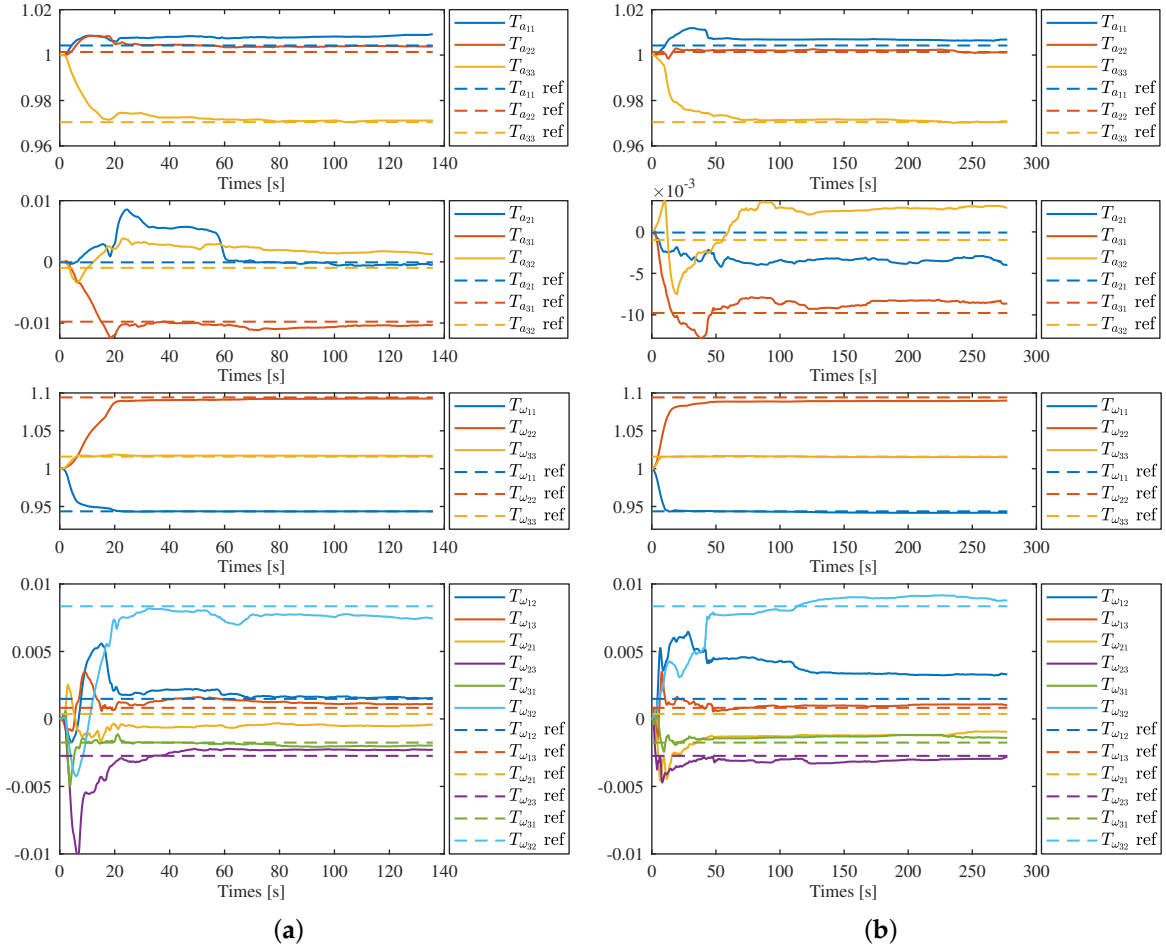

Figure 5. IMU intrinsics estimation in room 1 (**a**) and slider 1 (**b**). Solid lines are the estimation results and the dotted lines with the same color are the corresponding reference value.

It is not surprising for the results mentioned above, as many factors would affect the accuracy of the estimation results. Firstly, the VINS problem has the characteristics of high nonlinearity and local weak observability. To achieve full observability of the IMU intrinsic parameters as well as the measurement bias, motion with sufficient excitation was required. It is certain that this condition would not be always satisfied at any time in real applications. Therefore, the result shows fluctuation under insufficient excitation motion. Besides, the convergence speeds were affected by different motions, which shows a different convergence speed for each sequence. Secondly, as the accelerometer measures the acceleration and the gyroscope can measure the angular velocity directly, this also means that different conditions are required for the accelerometer intrinsics and gyroscope intrinsics to achieve observability. For the accelerometer, sufficient acceleration on each axis is required, and for the gyroscope, only sufficient rotation of the system is required. For a moving sensing system, keeping sufficient acceleration all the time is infeasible in real applications. In contrast, keeping sufficient rotation is a more easily satisfied condition. As a result, the convergence speed of gyroscope shows faster and more stable than that of accelerometer. Thirdly, the measurements of vision sensor and IMU are affected by random noise, which is coupled with the erros caused by inaccuracy intrinsics and bias, and further affects the accuracy of the estimation results. Generally, the measurement noise of gyroscopes is much smaller than that of accelerometers, which means the estimated results of gyroscopes are more accurate than accelerometers.

From Figures 4 and 5, it seems that the estimation results of intrinsic parameters located at the diagonal of matrix $\boldsymbol{T}_a$ and $\boldsymbol{T}_\omega$ are more stable and accurate than other parameters located at the lower

or upper triangle. This is because parameters of the lower or upper triangle themselves are much smaller than those of diagonal, and any small challenges would show large fluctuations in the figures. To further evaluate the estimate results, we take the estimation values at the end of each sequence as the final estimation results and calculate their mean value, standard deviation as well as the differences compared with the corresponding reference values. The statistics are shown in the Tables 1 and 2. We can see that: (1) the standard deviation of diagonal parameters and that of off-diagonal parameters do not show obvious differences; (2) The standard deviation of gyroscope intrinsics is much smaller than that of accelerometer intrinsics. In Table 1, the standard deviation of results was about 0.002. In contrast, five of nine parameters' standard deviations in Table 2 were less than 0.001. This means that the results of the gyroscope were more accurate, which concludes the analysis above. Generally, the axis misalignments of low-cost MEMS-based IMU was 0.5–1%, and the standard deviations of our estimation results was in the order of magnitude 10^{-3}. Therefore, we can conclude that the proposed method can works well with the low-cost MEMS-based IMU.

Table 1. Statistics of the estimation results of accelerometer intrinsics.

	$T_{a_{11}}$	$T_{a_{21}}$	$T_{a_{22}}$	$T_{a_{31}}$	$T_{a_{32}}$	$T_{a_{33}}$
Reference	1.0042	−0.0001	1.0014	−0.0098	−0.0010	0.9705
Mean	1.0060	−0.0019	1.0040	−0.0090	0.0006	0.9700
Standard deviations	0.0017	0.0024	0.0020	0.0010	0.0028	0.0021
|Mean-reference|	0.0018	0.0018	0.0026	0.0008	0.0016	0.0004

Table 2. Statistics of the estimation results of gyroscope intrinsics.

	$T_{\omega_{11}}$	$T_{\omega_{12}}$	$T_{\omega_{13}}$	$T_{\omega_{21}}$	$T_{\omega_{22}}$	$t_{\omega_{23}}$	$t_{\omega_{31}}$	$t_{\omega_{32}}$	$t_{\omega_{33}}$
Reference	0.9436	0.0015	0.0008	0.0004	1.0941	−0.0027	−0.0018	0.0083	1.0159
Mean	0.9416	0.0023	0.0009	−0.0005	1.0919	−0.0012	−0.0015	0.0064	1.0160
Standard deviations	0.0009	0.0009	0.0004	0.0006	0.0018	0.0022	0.0004	0.0030	0.0012
|Mean-reference|	0.0020	0.0008	0.0000	0.0009	0.0022	0.0015	0.0003	0.0019	0.0002

4.1.2. IMU Intrinsics Estimation with G-Ensitivity

As the g-sensitivity coefficients are unknown, here we only show the results of sequence of corridor 1 and magistrale 1 in detail, as in Figure 6a,b. Comparing these two figures with the estimation results without g-sensitivity showed in Figure 4a,b, we can find that the reuslts of $\boldsymbol{T}_a$ and $\boldsymbol{T}_\omega$ are nearly same, regardless of with or without g-sensivity estimation. The results of g-sensivity $\boldsymbol{A}_\omega$ do not show that they are very stable. However, this does not mean that we obtained the wrong estimations, or that the calibration of g-sensitivity is totally failed. We would further discuss this in the next section with the quantitative results of VINS to indirectly evaluation to g-sensitivity estimation.

4.2. Quantitative Evaluation

The goal of the IMU calibration is to estimate the IMU intrinsics and compensate the measurements of IMU, and to improve the performance of VINS. Therefore, the accuracy of the IMU intrinsics calibration would directly affects the results of motion estimation. In this section, we quantify the accuracy of the estimation result of system pose with absolute trajectory error (ATE) and relative pose error (RPE) [53]. The RPE is only available for the sequences of rooms 1–6, as for other sequences only accurate pose ground truth at the start and end are available. For comparison, we also ran the VINS-Mono with the same camera image sequences and configuration (IMU noise, camera intrinsics, etc.), but with the precalibrated IMU measurements provided by the dataset. We did not run VINS-Mono directly with the raw IMU measurements as it fails after a few seconds on all sequences. For each sequence, we run the proposed algorithm and the VINS-Mono five times. In each run of the proposed algorithm, the IMU intrinsic parameters estimated online are saved after each nonlinear optimization step and used as the initial values of the next run. The median values obtained

by the five times running were used as the final result. The trajectory estimation results of sequence rooms 3 and 5 are showed in Figure 7. To simplify the notation, in the following figures and tables, we use VINS to denote the results of VINS-Mono, our results and our results with $\boldsymbol{A}_\omega$ to denote the results of our proposed method without and with g-sensitivity estimation, respectively.

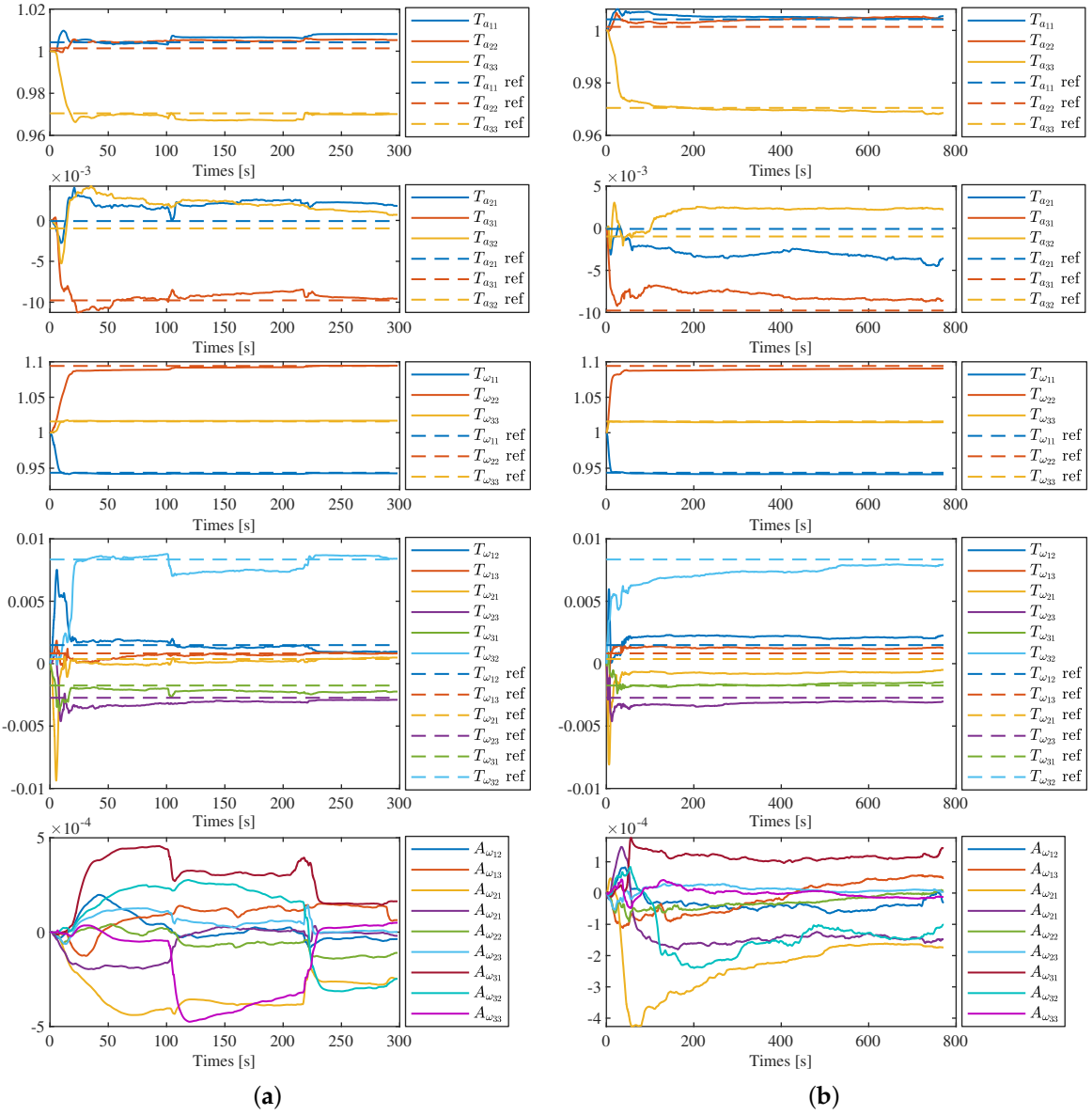

Figure 6. IMU intrinsics estimation with g-sensitivity in corridor 1 (**a**) and magistrale1 (**b**). Solid lines are the estimation results and the dot lines with the same color are the corresponding reference value.

The root-mean-square-error (RMSE) ATE of all sequences is shown in Table 3. Compared the results of our method without g-sensitivity estimation (column 3) against those of VINS-Mono (column 2), 13/20 of the our results method show improvement performance, with the rest of results offering near performance to those of VINS-Mono except the sequence magistrales 3 and 6. What interested us is that, although the estimation results of $T_{a_{21}}$ and $T_{a_{32}}$ in sequence magistrale 1 and slider 1 show larger differences compared to the reference values, the ATE of these two sequences show obvious improvement. This means there exist some errors in the precalibrated reference values or the intrinsic parameter values vary with time, as we mentioned in Section 1. Therefore, our VINS

implementaion with online IMU self-calibration shows superior performance compared against the result of VINS-Mono with offline precalibrated IMU measurements.

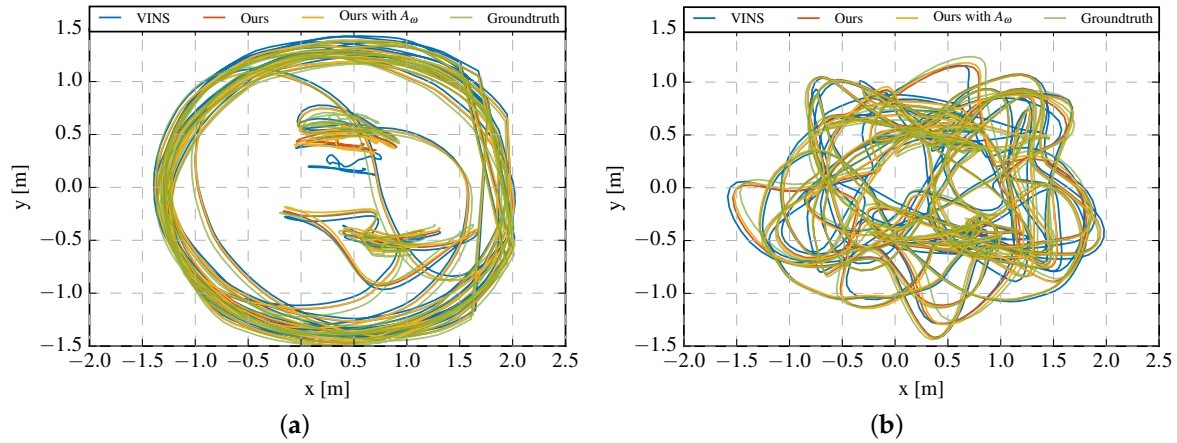

Figure 7. Trajectory of (**a**) rooms 3 and (**b**) room 5.

Now we further focus on the ATE results of our method with g-sensitivity estimation (column 4). 9/20 of the results show best performance between the three algorithms, most of which show more than 50% improvements. However, some of the results show worse performance compared to the estimation results without g-sensitivity calibration. This also concludes our discussion in Section 3.7 that online g-sensitivity calibration is a difficult task and we would not always get a better performance with g-sensitivity estimation.

Table 3. Root-mean-square-error (RMSE) absolute trajectory error (ATE) comparison between the proposed method with raw inertial measurement unit (IMU) measurements and monocular visual-inertial system (VINS-Mono) with precalibrated IMU measurements in all of the sequences. Downarrow means better performance compared to VINS-Mono. The bold values indicate the best results.

Sequence	VINS	Ours	Ours with A_ω	Length (m)
corridor1	0.618	**0.453** ↓	0.851	305
corridor2	1.174	1.157 ↓	**0.936** ↓	322
corridor3	1.309	0.797 ↓	**0.447** ↓	300
corridor4	0.309	0.195 ↓	**0.136** ↓	114
corridor5	0.674	**0.547** ↓	0.647 ↓	270
magistrale1	2.385	2.043 ↓	**1.072** ↓	918
magistrale2	3.205	**3.105** ↓	3.132 ↓	561
magistrale3	**0.358**	0.521	1.035	566
magistrale4	4.443	**3.919** ↓	4.223 ↓	688
magistrale5	**0.542**	0.603	0.738	458
magistrale6	2.078	2.825	**1.210** ↓	771
room1	**0.089**	0.095	0.094	146
room2	**0.048**	0.051	0.049	142
room3	0.145	**0.076** ↓	0.084 ↓	135
room4	**0.042**	0.048	0.052	68
room5	0.196	0.049 ↓	**0.043** ↓	131
room6	0.059	0.057 ↓	**0.048** ↓	67
slides1	0.517	0.273 ↓	**0.193** ↓	289
slides2	**1.007**	1.047	1.209	299
slides3	1.005	0.781 ↓	**0.764** ↓	383

Figure 8 shows the overall relative pose error (RTE) in room1–room6, caculated by the toolbox [54]. In Figure 8a, we can find obvious improvements compared to VINS-Mono. In contrast, nearly same relative orientation errors are showed in Figure 8b.

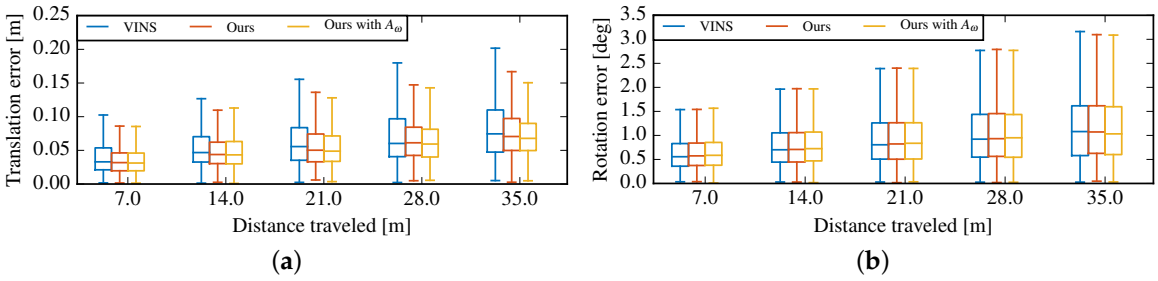

Figure 8. Overall relative pose error in rooms 1–6. (**a**) Relative translation error; (**b**) Relative orientation error

4.3. Runtime Evaluation

The proposed online IMU self-calibration method adds additional states to the system state vector, and estimates the optimal value of these states based on optimization method. As a result, online calibration would consume more computation resource. Figure 9 shows the average processing time per frame on each sequence. All experiments were performed using a computer equipped with an Intel Core(TM) i7-7700 CPU at 3.60 GHz and 8 GB RAM.

It is obvious that the proposed method consumes an additional 3–5 ms to estimate the IMU intrinsics. We further compute the average processing time for all sequences and the results are 27.7 ms, 31.8 ms, 33.5 ms, respectively. The processing time of our method without and with g-sensitivity estimation increase 14.9% and 20.9%, respectively, compared with VINS-Mono. Nevertheless, we can conclude that our method can run in real time.

Calibration process with g-sensitivity would introduce extra nine variables to the state space and consumes about additional 2 ms times the cost. Therefore, in practice applications, it is necessary to decide whether or not calibrating the g-sensitivity online, according to the IMU used and the trade-off between estimation accuracy and real time performance. Beside, the motion of the sensing system that are subjected should be considered, too. For example, if the general motion of the system is in low acceleration most of the time, g-sensitivity calibration may not significantly improve the performance of VINS.

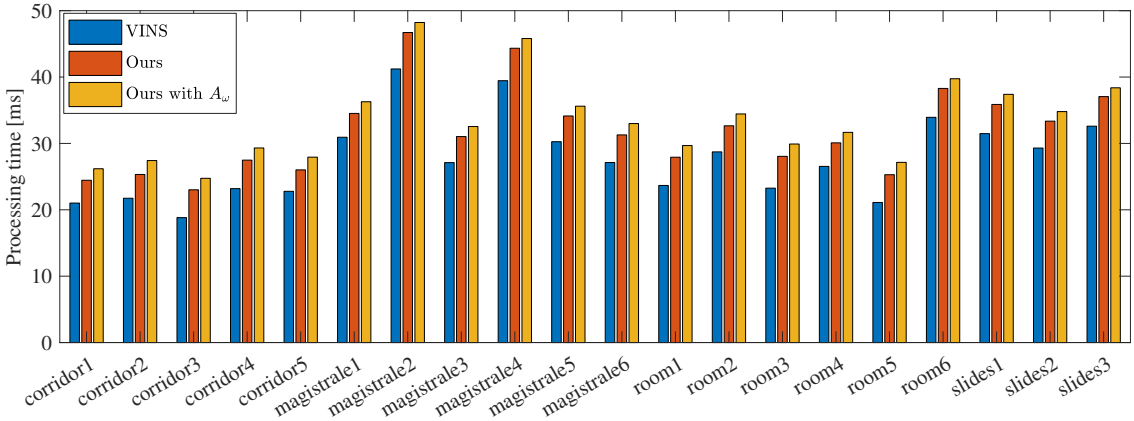

Figure 9. Average processing time per frame.

5. Conclusions

In this paper, we propose an online IMU self-calibration method for visual-inertial system with low-cost inertial sensors. The main advantage of our method is that it performs online IMU self-calibration based on optimization method in unknown environment, using only camera and inertial measurements, and without any knowledge about the structure of scene or any external equipment. Specifically, the estimated parameters include the axis misalignments and scale factors of the IMU, as well as the g-sensitivity of the gyroscope. We perform extensive evaluation on public TUM VI dataset, with and without g-sensitivity estimation, respectively. Experimental results verify that the proposed method is able to accurately calibrate all the considered parameters in real time, and show superior performance compared with the results of VINS with offline precalibrated IMU measurements. It should be noted that the calibration process with g-sensitivity would introduce extra nine variables to the state space and consume more computation resource. Therefore, in practical application, it is necessary to decide whether or not calibrating the g-sensitivity online, according to the IMU used and the trade-off between estimation accuracy and real time performance.

The proposed method is dedicated to low-cost IMUs. As a result, it is very suitable for the consumer grade mobile robots, drones and cellphones which are broadly equipped with low-cost MEMS-based IMUs. Moreover, our method would promote the rapid deployment of VINS in various applications base on these platforms, such as robot navigation, autonomous driving, AR and VR.

In this paper we adopt quaternion as global parameterization for rotations. In the future we will extend our method to work with the special orthogonal group SO(3), which would make it more easily be employed in existing SO(3)-based visual/visual-inertial frameworks.

Author Contributions: Conceptualization, Y.X.; funding acquisition, X.R. and X.Z. (Xiaoqing Zhu); methodology, Y.X.; project administration, X.R. and X.Z. (Xiaoqing Zhu); software, Y.X.; supervision, X.R.; writing—original draft, Y.X.; writing—review and editing, X.Z. (Xiaoqing Zhu), X.Z. (Xiaoping Zhang) and J.C.

Funding: This research has been funded by the National Natural Science Foundation of China (No. 61773027), Key Project of S and T Plan of Beijing Municipal Commission of Education (No. KZ201610005010) and Beijing Natural Science Foundation (4174083).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IMU	Inertial measurement unit
MEMS	Microelectro mechanical systems
AHRS	Attitude and heading reference system

Appendix A. Quaternion-Based IMU Preintegration

Appendix A.1. The Error-State Kinematics

This section details how to obtain the the error-state kinematic in Section 3.4. Our goal is to determine $\delta\dot{\boldsymbol{\alpha}}$, $\delta\dot{\boldsymbol{\beta}}$, $\delta\dot{\boldsymbol{\gamma}}$. For readable and simplicity purpose, we will drop the coordinate frame subscripts in the following equations, which would not cause ambiguous.

From Equations (14)–(16), we can get the true state kinematics of preintegration:

$$\dot{\boldsymbol{\alpha}} = \boldsymbol{\beta} \tag{A1}$$

$$\dot{\boldsymbol{\beta}} = \boldsymbol{R}\boldsymbol{T}_a(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) \tag{A2}$$

$$\dot{\boldsymbol{\gamma}} = \frac{1}{2}\boldsymbol{\gamma} \otimes \boldsymbol{T}_\omega\left(\boldsymbol{\omega}_m - \boldsymbol{A}_\omega T_a(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega\right). \tag{A3}$$

From Equations (19)–(21), the nominal-states kinematic is obtained as follows:

$$\dot{\hat{\boldsymbol{\alpha}}} = \hat{\boldsymbol{\beta}} \tag{A4}$$

$$\dot{\hat{\boldsymbol{\beta}}} = \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a(\boldsymbol{a}_m - \hat{\boldsymbol{b}}_a) \tag{A5}$$

$$\dot{\hat{\boldsymbol{\gamma}}} = \frac{1}{2}\hat{\boldsymbol{\gamma}} \otimes \hat{\boldsymbol{T}}_\omega(\boldsymbol{\omega}_m - \hat{\boldsymbol{A}}_\omega\hat{T}_a(\boldsymbol{a}_m - \hat{\boldsymbol{b}}_a) - \hat{\boldsymbol{b}}_\omega). \tag{A6}$$

A. The Translation Error

With Equations (A3) and (A6), it is easy to obtain:

$$\delta\dot{\boldsymbol{\alpha}} = \boldsymbol{\beta} - \hat{\boldsymbol{\beta}} = \delta\boldsymbol{\beta} \tag{A7}$$

B. The Linear Velocity Error

We start with the following relations:

$$\boldsymbol{R} = \hat{\boldsymbol{R}}(I + [\delta\boldsymbol{\theta}]_\times) + \mathcal{O}(\|\delta\boldsymbol{\theta}\|^2) \tag{A8}$$

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{T}_a(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) \\ &= \hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B + \delta\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \delta\hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B, \end{aligned} \tag{A9}$$

where $\hat{\boldsymbol{a}}_B = \boldsymbol{a}_m - \boldsymbol{b}_a$, $\delta\boldsymbol{a}_B = -\delta\boldsymbol{b}_a - \boldsymbol{n}_a$.

Substituting Equations (A8) and (A9) into Equations (A2) and (A5) yields:

$$\begin{aligned} \dot{\boldsymbol{\beta}} &\approx \hat{\boldsymbol{R}}(I + [\delta\boldsymbol{\theta}]_\times)(\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B + \delta\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \delta\hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B) \\ &= \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B + \hat{\boldsymbol{R}}\delta\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{R}}\delta\hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B + \hat{\boldsymbol{R}}[\delta\boldsymbol{\theta}]_\times\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B \\ \dot{\hat{\boldsymbol{\beta}}} &= \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B \\ \Rightarrow \delta\dot{\boldsymbol{\beta}} &= \dot{\boldsymbol{\beta}} - \dot{\hat{\boldsymbol{\beta}}} \\ &\approx \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta\boldsymbol{a}_B + \hat{\boldsymbol{R}}\delta\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{R}}[\delta\boldsymbol{\theta}]_\times\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B \\ &= -\hat{\boldsymbol{R}}[\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B]_\times\delta\boldsymbol{\theta} - \hat{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta\hat{\boldsymbol{b}}_a + \hat{\boldsymbol{R}}\delta\boldsymbol{T}_a\hat{\boldsymbol{a}}_B - \hat{\boldsymbol{R}}\boldsymbol{T}_a\boldsymbol{n}_a. \end{aligned} \tag{A10}$$

C. The Orientation Error

We start with the following relations

$$\begin{aligned} \boldsymbol{\omega} &= \boldsymbol{T}_\omega(\boldsymbol{\omega}_m - \boldsymbol{A}_\omega T_a(\boldsymbol{a}_m - \boldsymbol{b}_a - \boldsymbol{n}_a) - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega) \\ &= \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{\omega}}_B + \hat{\boldsymbol{T}}_\omega\delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{\omega}}_B + \delta\hat{\boldsymbol{T}}_\omega\delta\boldsymbol{\omega}_B, \end{aligned} \tag{A11}$$

where, $\hat{\boldsymbol{\omega}}_B = \boldsymbol{\omega}_m - \hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B - \hat{\boldsymbol{b}}_\omega$, $\delta\boldsymbol{\omega}_B = -\delta\boldsymbol{A}_\omega\hat{\boldsymbol{T}}_a\hat{\boldsymbol{a}}_B + \hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\delta\boldsymbol{b}_a + (\hat{\boldsymbol{A}}_\omega + \delta\boldsymbol{A}_\omega)\hat{\boldsymbol{T}}_a\boldsymbol{n}_a - \boldsymbol{A}_\omega\delta\hat{\boldsymbol{T}}_a(\hat{\boldsymbol{a}}_B - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) - \delta\boldsymbol{b}_\omega - \boldsymbol{n}_\omega$.

Substituting Equation (A11) in Equations (A3) and (A6) yields:

$$\begin{aligned}
&\dot{\gamma} = \frac{1}{2}\gamma \otimes \boldsymbol{\omega}, \dot{\hat{\gamma}} = \frac{1}{2}\hat{\gamma} \otimes \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \\
&\Rightarrow \dot{\gamma} = (\hat{\gamma} \dot{\otimes} \delta\gamma) = \frac{1}{2}(\hat{\gamma} \otimes \delta\gamma) \otimes \boldsymbol{\omega} \\
&\Rightarrow \dot{\hat{\gamma}} \otimes \delta\gamma + \hat{\gamma} \otimes \delta\dot{\gamma} = \frac{1}{2}(\hat{\gamma} \otimes \delta\gamma) \otimes \boldsymbol{\omega} \\
&\Rightarrow \frac{1}{2}\hat{\gamma} \otimes \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \otimes \delta\gamma + \hat{\gamma} \otimes \delta\dot{\gamma} = \frac{1}{2}(\hat{\gamma} \otimes \delta\gamma) \otimes \boldsymbol{\omega} \\
&\Rightarrow (\hat{\gamma})^{-1}(\frac{1}{2}\hat{\gamma} \otimes \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \otimes \delta\gamma + \hat{\gamma} \otimes \delta\dot{\gamma}) = (\hat{\gamma})^{-1}(\frac{1}{2}(\hat{\gamma} \otimes \delta\gamma) \otimes \boldsymbol{\omega}) \\
&\Rightarrow \frac{1}{2}\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \otimes \delta\gamma + \delta\dot{\gamma} = \frac{1}{2}\delta\gamma \otimes \boldsymbol{\omega} \\
&\Rightarrow \delta\dot{\gamma} = \frac{1}{2}\delta\gamma \otimes \boldsymbol{\omega} - \frac{1}{2}\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \otimes \delta\gamma.
\end{aligned}$$

Remember that $\delta\gamma = \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}$. Substitute it to the last equation above:

$$\begin{aligned}
\begin{bmatrix} 1 \\ \delta\dot{\boldsymbol{\theta}} \end{bmatrix} &= 2\delta\dot{\gamma} = \delta\gamma \otimes \boldsymbol{\omega} - \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B \otimes \delta\gamma \\
&= \boldsymbol{\Omega}_R(\boldsymbol{\omega})\delta\gamma - \boldsymbol{\Omega}_L(\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B)\delta\gamma \\
&= \begin{bmatrix} 0 & -(\boldsymbol{\omega} - \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B) \\ (\boldsymbol{\omega} - \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B) & -[\boldsymbol{\omega} + \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \end{bmatrix} \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} + \mathcal{O}(\|\delta\boldsymbol{\theta}\|^2) \\
&= \begin{bmatrix} 0 & -(\hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B) \\ (\hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B) & -[2\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B + \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \end{bmatrix} \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} + \mathcal{O}(\|\delta\boldsymbol{\theta}\|^2),
\end{aligned}$$

where, $\boldsymbol{\Omega}_L(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & [\boldsymbol{\omega}]_\times \end{bmatrix}$, $\boldsymbol{\Omega}_R(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \boldsymbol{\omega}^T \\ -\boldsymbol{\omega} & -[\boldsymbol{\omega}]_\times \end{bmatrix}$.

Now we obtain $\delta\dot{\boldsymbol{\theta}}$ as:

$$\begin{aligned}
\delta\dot{\boldsymbol{\theta}} &= (\hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B) - [2\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B + \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \frac{\delta\boldsymbol{\theta}}{2} + \mathcal{O}(\|\delta\boldsymbol{\theta}\|^2) \\
&\approx \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{\omega}_B + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B - [\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \delta\boldsymbol{\theta} \\
&= -[\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \delta\boldsymbol{\theta} + \delta\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B + \hat{\boldsymbol{T}}_\omega \big(- \delta\boldsymbol{A}_\omega \hat{\boldsymbol{T}}_a \hat{\boldsymbol{a}}_B + \hat{\boldsymbol{A}}_\omega \hat{\boldsymbol{T}}_a \delta\boldsymbol{b}_a \\
&\quad + (\hat{\boldsymbol{A}}_\omega + \delta\boldsymbol{A}_\omega)\hat{\boldsymbol{T}}_a \boldsymbol{n}_a - \boldsymbol{A}_\omega \delta\hat{\boldsymbol{T}}_a (\hat{\boldsymbol{a}}_B - \delta\boldsymbol{b}_a - \boldsymbol{n}_a) - \delta\boldsymbol{b}_\omega - \boldsymbol{n}_\omega \big) \\
&\approx -[\hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{\omega}}_B]_\times \delta\boldsymbol{\theta} + \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \hat{\boldsymbol{T}}_a \delta\boldsymbol{b}_a - \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{b}_\omega - \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \delta\boldsymbol{T}_a \hat{\boldsymbol{a}}_B \\
&\quad + \delta\boldsymbol{T}_\omega \hat{\boldsymbol{\omega}}_B - \hat{\boldsymbol{T}}_\omega \delta\boldsymbol{A}_\omega \hat{\boldsymbol{T}}_a \hat{\boldsymbol{a}}_B + \hat{\boldsymbol{T}}_\omega \hat{\boldsymbol{A}}_\omega \hat{\boldsymbol{T}}_a \boldsymbol{n}_a - \hat{\boldsymbol{T}}_\omega \boldsymbol{n}_\omega.
\end{aligned} \tag{A12}$$

Equations (A7), (A10) and (A12) form the error-state kinematic which is same as in Equations (22)–(24).

Appendix A.2. Mid-Point Integration for the IMU Preintegration

We perform the Mid-point numerical integration in our implementation code. This section shows the computation details.

As showed in the Figure 2, we assume that the i-th IMU measurement is obtained at the time instant of frame k, and j-th measurement at frame $k+1$. For a real sensor suite, there may be no IMU measurement at the time instant of a image frame is captured. In this case, linear interpolation method can be used to estimate a IMU measurement at the corresponding time instant. As showed in

Section 3.6, the $\boldsymbol{T}_a$, $\boldsymbol{T}_\omega$, $\boldsymbol{A}_\omega$ is also assumed constant in the timespan of sliding window, so we will not explicitly distinguish the $\boldsymbol{T}_{a,i}$, $\boldsymbol{T}_{\omega,i}$, $\boldsymbol{A}_{\omega,i}$ and $\boldsymbol{T}_{a,i+1}$, $\boldsymbol{T}_{\omega,i+1}$, $\boldsymbol{A}_{\omega,i+1}$ in the following equations.

A. Nominal-State Preintegration

Integrating Equations (19)–(21) is to get the nominal-state estimation:

$$\hat{\boldsymbol{\alpha}}_{i+1}^{b_k} = \hat{\boldsymbol{\alpha}}_i^{b_k} + \hat{\boldsymbol{\beta}}_{i+1}^{b_k}\delta t + \frac{1}{2}\hat{\boldsymbol{\gamma}}_i^{b_k}\hat{\boldsymbol{T}}_a(\frac{1}{2}(\boldsymbol{a}_{m_i} + \boldsymbol{a}_{m_{i+1}}) - \hat{\boldsymbol{b}}_a)\delta t^2 \tag{A13}$$

$$\hat{\boldsymbol{\beta}}_{i+1}^{b_k} = \hat{\boldsymbol{\beta}}_i^{b_k} + \hat{\boldsymbol{\gamma}}_i^{b_k}\hat{\boldsymbol{T}}_a(\frac{1}{2}(\boldsymbol{a}_{m_i} + \boldsymbol{a}_{m_{i+1}}) - \hat{\boldsymbol{b}}_a)\delta t \tag{A14}$$

$$\hat{\boldsymbol{\gamma}}_{i+1}^{b_k} = \hat{\boldsymbol{\gamma}}_i^{b_k} \otimes \hat{\boldsymbol{T}}_\omega(\frac{1}{2}(\boldsymbol{\omega}_{m_i} + \boldsymbol{\omega}_{m_{i+1}}) - \hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a(\frac{1}{2}(\boldsymbol{a}_{m_i} + \boldsymbol{a}_{m_{i+1}}) - \hat{\boldsymbol{b}}_{a_t}) - \hat{\boldsymbol{b}}_{\omega_t})\delta t, \tag{A15}$$

where δt is the time interval between t_i and t_{i+1}. At the beginning of integration, $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$, $\boldsymbol{\beta}_{b_{k+1}}^{b_k}$ is 0, and $\gamma_{b_{k+1}}^{b_k}$ is identity quaternion. The prorogation starts with $i = i$ and repeats step by step until $i + 1 = j$.

In the following we will detail how to obtain the error-state Jacobian and covariance using the Mid-point integration.

B. Error-State of Orientation Preintegration

Define $\bar{\boldsymbol{\omega}}_B = \frac{1}{2}(\hat{\boldsymbol{\omega}}_{B,i} + \hat{\boldsymbol{\omega}}_{B,i+1})$, $\bar{\boldsymbol{a}}_B = \frac{1}{2}(\hat{\boldsymbol{a}}_{B,i} + \hat{\boldsymbol{a}}_{B,i+1})$. Integration Equation (A12) in discrete time:

$$\begin{aligned}\delta\boldsymbol{\theta}_{i+1} &= \big(\boldsymbol{I} - [\hat{\boldsymbol{T}}_\omega\bar{\boldsymbol{\omega}}_B]_\times\delta t\big)\delta\boldsymbol{\theta}_i + \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\delta t\delta\boldsymbol{b}_{a,i} - \hat{\boldsymbol{T}}_\omega\delta t\delta\boldsymbol{b}_{\omega,i} - \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\delta\boldsymbol{T}_a\bar{\boldsymbol{a}}_B\delta t \\ &\quad + \delta\boldsymbol{T}_\omega\bar{\boldsymbol{a}}_B\delta t - \hat{\boldsymbol{T}}_\omega\delta\boldsymbol{A}_\omega\hat{\boldsymbol{T}}_a\bar{\boldsymbol{a}}_B\delta t + \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\delta t\boldsymbol{n}_a - \hat{\boldsymbol{T}}_\omega\delta t\boldsymbol{n}_\omega \\ &= \boldsymbol{f}_{33}\delta\boldsymbol{\theta}_i + \boldsymbol{f}_{34}\delta\boldsymbol{b}_{a,i} + \boldsymbol{f}_{35}\delta\boldsymbol{b}_{\omega,i} + \boldsymbol{f}_{36}\delta\boldsymbol{t}_{a,i} + \boldsymbol{f}_{37}\delta\boldsymbol{t}_{\omega,i} + \boldsymbol{f}_{38}\delta\boldsymbol{a}_{\omega,i} + \boldsymbol{n}_{31}\boldsymbol{n}_a + \boldsymbol{n}_{32}\boldsymbol{n}_\omega,\end{aligned} \tag{A16}$$

where,

$$\begin{aligned}&\boldsymbol{f}_{33} = \boldsymbol{I} - [\hat{\boldsymbol{T}}_\omega\bar{\boldsymbol{\omega}}_B]_\times\delta t,\ \boldsymbol{f}_{34} = \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\delta t,\ \boldsymbol{f}_{35} = \hat{\boldsymbol{T}}_\omega\delta t,\ \boldsymbol{f}_{36} = -\hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega[\bar{\boldsymbol{a}}_B]_T\delta t \\ &\boldsymbol{f}_{37} = [\bar{\boldsymbol{a}}_B]_T\delta t,\ \boldsymbol{f}_{38} = -\hat{\boldsymbol{T}}_\omega[\hat{\boldsymbol{T}}_a\bar{\boldsymbol{a}}_B]_T\delta t,\ \boldsymbol{n}_{31} = \hat{\boldsymbol{T}}_\omega\hat{\boldsymbol{A}}_\omega\hat{\boldsymbol{T}}_a\delta t,\ \boldsymbol{n}_{32} = -\hat{\boldsymbol{T}}_\omega\delta t.\end{aligned}$$

C. Error-State of Velocity Preintegration

Define

$$\begin{aligned}\boldsymbol{R}_{a0} &= \frac{1}{2}\hat{\boldsymbol{R}}_i[\hat{\boldsymbol{T}}_a\boldsymbol{a}_{B,i}]_\times, \boldsymbol{R}_{a1} = \frac{1}{2}\hat{\boldsymbol{R}}_{i+1}[\hat{\boldsymbol{T}}_a\boldsymbol{a}_{B,i+1}]_\times \\ \bar{\boldsymbol{R}} &= \frac{1}{2}(\hat{\boldsymbol{R}}_i + \hat{\boldsymbol{R}}_{i+1}) \\ \boldsymbol{R}_{a_T} &= \frac{1}{2}(\hat{\boldsymbol{R}}_i[\boldsymbol{a}_{B,i}]_T + \hat{\boldsymbol{R}}_{i+1}[\boldsymbol{a}_{B,i+1]_T}).\end{aligned}$$

Integration Equation (A10) in discrete time:

$$\begin{aligned}\delta\boldsymbol{\beta}_{i+1} &= \delta\boldsymbol{\beta}_i - (\boldsymbol{R}_{a0}\delta\boldsymbol{\theta}_i + \boldsymbol{R}_{a1}\delta\boldsymbol{\theta}_{i+1})\delta t - \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta t\delta\hat{\boldsymbol{b}}_{a,i} + \boldsymbol{R}_{a_T}\delta t\delta\boldsymbol{t}_{a,i} - \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta t\boldsymbol{n}_a \\ &= \delta\boldsymbol{\beta}_i - (\boldsymbol{R}_{a0}\delta\boldsymbol{\theta}_i + \boldsymbol{R}_{a1}\big(\boldsymbol{f}_{33}\delta\boldsymbol{\theta}_i + \boldsymbol{f}_{34}\delta\boldsymbol{b}_{a,i} + \boldsymbol{f}_{35}\delta\boldsymbol{b}_{\omega,i} + \boldsymbol{f}_{36}\delta\boldsymbol{t}_{a,i} + \boldsymbol{f}_{37}\delta\boldsymbol{t}_{\omega,i} + \boldsymbol{f}_{38}\delta\boldsymbol{a}_{\omega,i} \\ &\quad + \boldsymbol{n}_{31}\boldsymbol{n}_a + \boldsymbol{n}_{32}\boldsymbol{n}_\omega\big))\delta t - \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta t\delta\hat{\boldsymbol{b}}_a + \boldsymbol{R}_{a_T}\delta t\delta\boldsymbol{t}_a - \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a\delta t\boldsymbol{n}_a \\ &= \delta\boldsymbol{\beta}_i + \boldsymbol{f}_{23}\delta\boldsymbol{\theta}_i + \boldsymbol{f}_{24}\delta\boldsymbol{b}_{a,i} + \boldsymbol{f}_{25}\delta\boldsymbol{b}_{\omega,i} + \boldsymbol{f}_{26}\delta\boldsymbol{t}_{a,i} + \boldsymbol{f}_{27}\delta\boldsymbol{t}_{\omega,i} + \boldsymbol{f}_{28}\delta\boldsymbol{a}_{\omega,i} + \boldsymbol{n}_{21}\boldsymbol{n}_a + \boldsymbol{n}_{22}\boldsymbol{n}_\omega,\end{aligned} \tag{A17}$$

where,

$$\begin{aligned}&\boldsymbol{f}_{23} = -(\boldsymbol{R}_{a0} + \boldsymbol{R}_{a1}\boldsymbol{f}_{33})\delta t, \boldsymbol{f}_{24} = -(\boldsymbol{R}_{a1}\boldsymbol{f}_{33} + \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a)\delta t, \boldsymbol{f}_{25} = -\boldsymbol{R}_{a1}\boldsymbol{f}_{35}\delta t, \\ &\boldsymbol{f}_{26} = (-\boldsymbol{R}_{a1}\boldsymbol{f}_{36} + \boldsymbol{R}_{a_T})\delta t, \boldsymbol{f}_{27} = -\boldsymbol{R}_{a1}\boldsymbol{f}_{37}\delta t,\ \boldsymbol{f}_{28} = -\boldsymbol{R}_{a1}\boldsymbol{f}_{38}\delta t, \\ &\boldsymbol{n}_{21} = -(\boldsymbol{R}_{a1}\boldsymbol{n}_{21} + \bar{\boldsymbol{R}}\hat{\boldsymbol{T}}_a)\delta t,\ \boldsymbol{n}_{22} = -\boldsymbol{R}_{a1}\boldsymbol{n}_{22}\delta t.\end{aligned}$$

D. Error-State of Translation Preintegration

$$\begin{aligned}\delta\boldsymbol{\alpha}_{i+1} &= \delta\boldsymbol{\alpha}_i + \delta\boldsymbol{\beta}_i\delta t + \frac{1}{2}\delta\dot{\boldsymbol{\beta}}_i\delta t^2 \\ &= \delta\boldsymbol{\alpha}_i + \delta\boldsymbol{\beta}_i\delta t + \frac{\delta t}{2}\big(\boldsymbol{f}_{23}\delta\boldsymbol{\theta}_i + \boldsymbol{f}_{24}\delta\boldsymbol{b}_{a,i} + \boldsymbol{f}_{25}\delta\boldsymbol{b}_{\omega,i} + \boldsymbol{f}_{26}\delta\boldsymbol{t}_{a,i} \\ &\qquad + \boldsymbol{f}_{27}\delta\boldsymbol{t}_{\omega,i} + \boldsymbol{f}_{28}\delta\boldsymbol{a}_{\omega,i} + \boldsymbol{n}_{21}\boldsymbol{n}_a + \boldsymbol{n}_{22}\boldsymbol{n}_\omega\big).\end{aligned} \tag{A18}$$

Equations (A16)–(A18) can be represent in matrix form:

$$\begin{bmatrix}\boldsymbol{\alpha}_{i+1}\\ \boldsymbol{\beta}_{i+1}\\ \boldsymbol{\gamma}_{i+1}\\ \boldsymbol{b}_{a,i+1}\\ \boldsymbol{b}_{\omega,i+1}\\ \boldsymbol{t}_{a,i+1}\\ \boldsymbol{t}_{\omega,i+1}\\ \boldsymbol{a}_{\omega,i+1}\end{bmatrix} = \underbrace{\begin{bmatrix}\boldsymbol{I}_3 & \frac{\delta t}{2}\boldsymbol{f}_{23} & \boldsymbol{I}_3\delta t & \frac{\delta t}{2}\boldsymbol{f}_{24} & \frac{\delta t}{2}\boldsymbol{f}_{25} & \frac{\delta t}{2}\boldsymbol{f}_{26} & \frac{\delta t}{2}\boldsymbol{f}_{27} & \frac{\delta t}{2}\boldsymbol{f}_{28}\\ \boldsymbol{0} & \boldsymbol{f}_{22} & \boldsymbol{I}_3 & \boldsymbol{f}_{24} & \boldsymbol{f}_{25} & \boldsymbol{f}_{26} & \boldsymbol{f}_{27} & \boldsymbol{f}_{28}\\ \boldsymbol{0} & \boldsymbol{f}_{32} & \boldsymbol{0} & \boldsymbol{f}_{34} & \boldsymbol{f}_{35} & \boldsymbol{f}_{36} & \boldsymbol{f}_{37} & \boldsymbol{f}_{38}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_3 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_3 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_6 & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_9 & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_9\end{bmatrix}}_{\boldsymbol{F}_{m,i}} \begin{bmatrix}\boldsymbol{\alpha}_i\\ \boldsymbol{\beta}_i\\ \boldsymbol{\gamma}_i\\ \boldsymbol{b}_{a,i}\\ \boldsymbol{b}_{\omega,i}\\ \boldsymbol{t}_{a,i}\\ \boldsymbol{t}_{\omega,i}\\ \boldsymbol{a}_{\omega,i}\end{bmatrix} + \underbrace{\begin{bmatrix}\frac{\delta t}{2}\boldsymbol{n}_{21} & \frac{\delta t}{2}\boldsymbol{n}_{22} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{n}_{21} & \boldsymbol{n}_{22} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{n}_{31} & \boldsymbol{n}_{32} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}}_{\boldsymbol{G}_{m,i}} \begin{bmatrix}\boldsymbol{n}_a\\ \boldsymbol{n}_\omega\\ \boldsymbol{n}_{ba}\\ \boldsymbol{n}_{b_\omega}\end{bmatrix}. \tag{A19}$$

Again, the Jacobian and covariance matrix can propagate recursively:

$$\boldsymbol{J}_{i+1} = \boldsymbol{F}_{m,i}\boldsymbol{J}_i \tag{A20}$$

$$\boldsymbol{P}_{i+1} = \boldsymbol{F}_{m,i}\boldsymbol{P}_i\boldsymbol{F}_{m,i}^T + \boldsymbol{G}_{m,i}\boldsymbol{Q}\boldsymbol{G}_{m,i}^T. \tag{A21}$$

References

1. Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 15–22, doi:10.1109/ICRA.2014.6906584. [CrossRef]
2. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625. [CrossRef] [PubMed]
3. Howard, A. Real-time stereo visual odometry for autonomous ground vehicles. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 3946–3952, doi:10.1109/IROS.2008.4651147. [CrossRef]
4. Wang, R.; Schworer, M.; Cremers, D. Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 3923–3931, doi:10.1109/ICCV.2017.421. [CrossRef]
5. Kerl, C.; Sturm, J.; Cremers, D. Dense visual SLAM for RGB-D cameras. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2100–2106, doi:10.1109/IROS.2013.6696650. [CrossRef]

6. Kerl, C.; Sturm, J.; Cremers, D. Robust odometry estimation for RGB-D cameras. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3748–3754, doi:10.1109/ICRA.2013.6631104. [CrossRef]
7. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Robot. Res.* **2015**, *34*, 314–334, doi:10.1177/0278364914554813. [CrossRef]
8. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020, doi:10.1109/TRO.2018.2853729. [CrossRef]
9. Mourikis, A.I.; Roumeliotis, S.I. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3565–3572, doi:10.1109/ROBOT.2007.364024. [CrossRef]
10. Li, M.; Mourikis, A.I. High-precision, consistent EKF-based visual-inertial odometry. *Int. J. Robot. Res.* **2013**, *32*, 690–711, doi:10.1177/0278364913481251. [CrossRef]
11. Weiss, S.; Siegwart, R. Real-time metric state estimation for modular vision-inertial systems. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; Volume 231855, pp. 4531–4537, doi:10.1109/ICRA.2011.5979982. [CrossRef]
12. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to MAV navigation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 3923–3929, doi:10.1109/IROS.2013.6696917. [CrossRef]
13. Li, M.; Yu, H.; Zheng, X.; Mourikis, A.I. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 409–416, doi:10.1109/ICRA.2014.6906889. [CrossRef]
14. Wu, K.; Ahmed, A.; Georgiou, G.; Roumeliotis, S. A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices. In Proceedings of the Robotics: Science and Systems XI, Rome, Italy, 13–17 July 2015, doi:10.15607/RSS.2015.XI.008. [CrossRef]
15. Paul, M.K.; Wu, K.; Hesch, J.A.; Nerurkar, E.D.; Roumeliotis, S.I. A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 165–172, doi:10.1109/ICRA.2017.7989022. [CrossRef]
16. Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R. Robust visual inertial odometry using a direct EKF-based approach. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; Volume 2015, pp. 298–304, doi:10.1109/IROS.2015.7353389. [CrossRef]
17. Leutenegger, S.; Furgale, P.; Rabaud, V.; Chli, M.; Konolige, K.; Siegwart, R. Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization. In Proceedings of the Robotics: Science and Systems IX, Berlin, Germany, 24–28 June 2013, doi:10.15607/RSS.2013.IX.037. [CrossRef]
18. Mur-Artal, R.; Tardos, J.D. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robot. Autom. Lett.* **2017**, *2*, 796–803, doi:10.1109/LRA.2017.2653359. [CrossRef]
19. Usenko, V.; Engel, J.; Stuckler, J.; Cremers, D. Direct visual-inertial odometry with stereo cameras. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1885–1892, doi:10.1109/ICRA.2016.7487335. [CrossRef]
20. Kasyanov, A.; Engelmann, F.; Stuckler, J.; Leibe, B. Keyframe-based visual-inertial online SLAM with relocalization. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 6662–6669, doi:10.1109/IROS.2017.8206581. [CrossRef]
21. Von Stumberg, L.; Usenko, V.; Cremers, D. Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2510–2517, doi:10.1109/ICRA.2018.8462905. [CrossRef]
22. Titterton, D.; Weston, J. *Strapdown Inertial Navigation Technology*, 2nd ed.; Institution of Engineering and Technology: Stevenage, UK, 2004.
23. Tedaldi, D.; Pretto, A.; Menegatti, E. A robust and easy to implement method for IMU calibration without external equipments. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 3042–3049, doi:10.1109/ICRA.2014.6907297. [CrossRef]

24. Rogers, R.M. *Applied Mathematics in Integrated Navigation Systems*, 2nd ed.; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2007, doi:10.2514/4.861598.
25. Chatfield, A.B. *Fundamentals of High Accuracy Inertial Navigation*; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 1997, doi:10.2514/4.866463.
26. Hall, J.J.; Williams, R.L. Case study: Inertial measurement unit calibration platform. *J. Field Robot.* **2000**, *17*, 623–632. [CrossRef]
27. Syed, Z.F.; Aggarwal, P.; Goodall, C.; Niu, X.; Elsheimy, N. A new multi-position calibration method for MEMS inertial navigation systems. *Meas. Sci. Technol.* **2007**, *18*, 1897–1907. [CrossRef]
28. Kim, A.; Golnaraghi, M.F. Initial calibration of an inertial measurement unit using an optical position tracking system. In Proceedings of the Position Location and Navigation Symposium, Monterey, CA, USA, 26–29 April 2004.
29. Nebot, E.; Durrant-Whyte, H. Initial Calibration and Alignment of Low-Cost Inertial Navigation Units for Land Vehicle Applications. *J. Robot. Syst.* **1999**, *16*, 81–92. [CrossRef]
30. Lotters, J.C.; Schipper, J.; Veltink, P.H.; Olthuis, W.; Bergveld, P. Procedure for in-use calibration of triaxial accelerometers in medical applications. *Sens. Actuators A Phys.* **1998**, *68*, 221–228. [CrossRef]
31. Fong, W.T.; Ong, S.K.; Nee, A.Y.C. Methods for in-field user calibration of an inertial measurement unit without external equipment. *Meas. Sci. Technol.* **2008**, *19*, 085202. [CrossRef]
32. Hwangbo, M.; Kim, J.S.; Kanade, T. IMU Self-Calibration Using Factorization. *IEEE Trans. Robot.* **2013**, *29*, 493–507. [CrossRef]
33. Rehder, J.; Nikolic, J.; Schneider, T.; Hinzmann, T.; Siegwart, R. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 4304–4311, doi:10.1109/ICRA.2016.7487628. [CrossRef]
34. Furgale, P.; Barfoot, T.D.; Sibley, G. Continuous-time batch estimation using temporal basis functions. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; Volume 34, pp. 2088–2095, doi:10.1109/ICRA.2012.6225005. [CrossRef]
35. Lupton, T.; Sukkarieh, S. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Trans. Robot.* **2012**, *28*, 61–76, doi:10.1109/TRO.2011.2170332. [CrossRef]
36. Shen, S.; Michael, N.; Kumar, V. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 5303–5310, doi:10.1109/ICRA.2015.7139939. [CrossRef]
37. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation. In Proceedings of the Robotics: Science and Systems XI, Rome, Italy, 13–17 July 2015, doi:10.15607/RSS.2015.XI.006. [CrossRef]
38. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21, doi:10.1109/TRO.2016.2597321. [CrossRef]
39. Qin, T.; Shen, S. Robust initialization of monocular visual-inertial estimation on aerial robots. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 4225–4232, doi:10.1109/IROS.2017.8206284. [CrossRef]
40. Madyastha, V.; Ravindra, V.; Mallikarjunan, S.; Goyal, A. Extended Kalman Filter vs. Error State Kalman Filter for Aircraft Attitude Estimation. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Portland, OR, USA, 8–11 August 2011; pp. 6615–6638.
41. Sola, J. Quaternion kinematics for the error-state Kalman filter. *arXiv* **2017**, arXiv:1711.02508.
42. Shi, J.; Tomasi, C. Good features to track. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-94), Seattle, WA, USA, 21–23 June 1994; pp. 593–600, doi:10.1109/CVPR.1994.323794. [CrossRef]
43. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81), Vancouver, BC, Canada, 24–28 August 1981; pp. 674–679.
44. Agarwal, S.; Mierle, K. Ceres Solver. Available online: http://ceres-solver.org (accessed on 2 November 2018).
45. Carlevarisbianco, N.; Ushani, A.K.; Eustice, R.M. University of Michigan North Campus long-term vision and lidar dataset. *Int. J. Robot. Res.* **2016**, *35*, 1023–1035. [CrossRef]

46. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163, doi:10.1177/0278364915620033. [CrossRef]
47. Pfrommer, B.; Sanket, N.; Daniilidis, K.; Cleveland, J. PennCOSYVIO: A challenging Visual Inertial Odometry benchmark. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3847–3854, doi:10.1109/ICRA.2017.7989443. [CrossRef]
48. Majdik, A.; Till, C.; Scaramuzza, D. The Zurich urban micro aerial vehicle dataset. *Int. J. Robot. Res.* **2017**, *36*, 269–273. [CrossRef]
49. Schubert, D.; Goll, T.; Demmel, N.; Usenko, V.; Stuckler, J.; Cremers, D. The TUM VI Benchmark for Evaluating Visual-Inertial Odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1680–1687, doi:10.1109/IROS.2018.8593419. [CrossRef]
50. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163, doi:10.1109/TRO.2015.2463671. [CrossRef]
51. Mur-Artal, R.; Tardos, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
52. Mu, X.; Chen, J.; Zhou, Z.; Leng, Z.; Fan, L. Accurate Initial State Estimation in a Monocular Visual-Inertial SLAM System. *Sensors* **2018**, *18*, 506, doi:10.3390/s18020506. [CrossRef]
53. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, 7–12 October 2012; pp. 573–580.
54. Zhang, Z.; Scaramuzza, D. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.