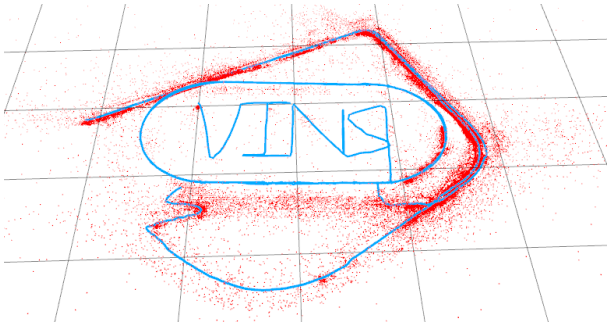
VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator

Tong Qin, Peiliang Li, and Shaojie Shen

***Abstract*—A monocular visual-inertial system (VINS), consisting of a camera and a low-cost inertial measurement unit (IMU), forms the minimum sensor suite for metric six degrees-of-freedom (DOF) state estimation. However, the lack of direct distance measurement poses significant challenges in terms of IMU processing, estimator initialization, extrinsic calibration, and nonlinear optimization. In this work, we present VINS-Mono: a robust and versatile monocular visual-inertial state estimator. Our approach starts with a robust procedure for estimator initialization and failure recovery. A tightly-coupled, nonlinear optimization-based method is used to obtain high accuracy visual-inertial odometry by fusing pre-integrated IMU measurements and feature observations. A loop detection module, in combination with our tightly-coupled formulation, enables relocalization with minimum computation overhead. We additionally perform four degrees-of-freedom pose graph optimization to enforce global consistency. We validate the performance of our system on public datasets and real-world experiments and compare against other state-of-the-art algorithms. We also perform onboard closed-loop autonomous flight on the MAV platform and port the algorithm to an iOS-based demonstration. We highlight that the proposed work is a reliable, complete, and versatile system that is applicable for different applications that require high accuracy localization. We open source our implementations for both PCs[1] and iOS mobile devices[2].**

***Index Terms*—Monocular visual-inertial systems, state estimation, sensor fusion, simultaneous localization and mapping**



(a) Trajectory (blue) and feature locations (red)

(b) Trajectory overlaid with Google Map for visual comparison

Fig. 1. Outdoor experimental results of the proposed monocular visual-inertial state estimator. Data is collected by a hand-held monocular camera-IMU setup under normal walking condition. It includes two complete circles inside the field and two semicircles on the nearby driveway. Total trajectory length is 2.5 km. A video of the experiment can be found in the multimedia attachment.

I. Introduction

STATE estimation is undoubtedly the most fundamental module for a wide range of applications, such as robotic navigation, autonomous driving, virtual reality (VR), and augmented reality (AR). Approaches that use only a monocular camera have gained significant interests by the community due to their small size, low-cost, and easy hardware setup [1]–[5]. However, monocular vision-only systems are incapable of recovering the metric scale, therefore limiting their usage in real-world robotic applications. Recently, we see a growing trend of assisting the monocular vision system with a low-cost inertial measurement unit (IMU). The primary advantage of this monocular visual-inertial system (VINS) is to have the metric scale, as well as roll and pitch angles, all observable. This enables navigation tasks that require metric state estimates. In addition, the integration of IMU measurements can dramatically improve motion tracking performance by bridging the gap between losses of visual tracks due to illumination change, texture-less area, or motion blur. In fact, the monocular VINS not only widely available on mobile robots, drones, and mobile devices, it is also the minimum sensor setup for sufficient self and environmental perception.

However, all these advantages come with a price. For monocular VINS, it is well known that acceleration excitation is needed to make metric scale observable. This implies that monocular VINS estimators cannot start from a stationary condition, but rather launch from an unknown moving state. Also recognizing the fact that visual-inertial systems are highly nonlinear, we see significant challenges in terms of estimator initialization. The existence of two sensors also makes camera-IMU extrinsic calibration critical. Finally, in order to eliminate long-term drift within an acceptable processing window, a complete system that includes visual-inertial odometry, loop detection, relocalization, and global optimization has to be developed.

T. Qin, P. Li, and S. Shen are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology. e-mail: {tqinab, pliap}@connect.ust.hk, eeshaojie@ust.hk

[1]https://github.com/HKUST-Aerial-Robotics/VINS-Mono

[2]https://github.com/HKUST-Aerial-Robotics/VINS-Mobile

To address all these issues, we propose VINS-Mono, a robust and versatile monocular visual-inertial state estimator. Our solution starts with on-the-fly estimator initialization. The same initialization module is also used for failure recovery. The core of our solution is a robust monocular visual-inertial odometry (VIO) based on tightly-coupled sliding window nonlinear optimization. The monocular VIO module not only provides accurate local pose, velocity, and orientation estimates, it also performs camera-IMU extrinsic calibration and IMU biases correction in an online fashion. Loops are detected using DBoW2 [6]. Relocalization is done in a tightly-coupled setting by feature-level fusion with the monocular VIO. This enables robust and accurate relocalization with minimum computation overhead. Finally, geometrically verified loops are added into a pose graph, and thanks to the observable roll and pitch angles from the monocular VIO, a four degrees-of-freedom (DOF) pose graph is performed to ensure global consistency.

VINS-Mono combines and improves the our previous works on monocular visual-inertial fusion [7]–[10]. It is built on top our tightly-coupled, optimization-based formulation for monocular VIO [7], [8], and incorporates the improved initialization procedure introduced in [9]. The first attempt of porting to mobile devices was given in [10]. Further improvements of VINS-Mono comparing to our previous works include improved IMU pre-integration with bias correction, tightly-coupled relocalization, global pose graph optimization, extensive experimental evaluation, and a robust and versatile open source implementation.

The whole system is complete and easy-to-use. It has been successfully applied to small-scale AR scenarios, medium-scale drone navigation, and large-scale state estimation tasks. Superior performance has been shown against other state-of-the-art methods. To this end, we summarize our contributions as follow:

- A robust initialization procedure that is able to bootstrap the system from unknown initial states.
- A tightly-coupled, optimization-based monocular visual-inertial odometry with camera-IMU extrinsic calibration and IMU bias estimation.
- Online loop detection and tightly-coupled relocalization.
- Four DOF global pose graph optimization.
- Real-time performance demonstration for drone navigation, large-scale localization, and mobile AR applications.
- Open-source release for both the PC version that is fully integrated with ROS, as well as the iOS version that runs on iPhone6s or above.

The rest of the paper is structured as follows. In Sect. II, we discuss relevant literature. We give an overview of the complete system pipeline in Sect. III. Preprocessing steps for both visual and pre-integrated IMU measurements are presented in Sect. IV. In Sect. V, we discuss the estimator initialization procedure. A tightly-coupled, self-calibrating, nonlinear optimization-based monocular VIO, is presented in Sect. VI. Tightly-coupled relocalization and global pose graph optimization are presented in Sect. VII and Sect. VIII respectively. Implementation details and experimental results are shown in Sect. IX. Finally, the paper is concluded with a discussion and possible future research directions in Sect. X.

II. Related Work

Scholarly works on monocular vision-based state estimation/odometry/SLAM are extensive. Noticeable approaches include PTAM [1], SVO [2], LSD-SLAM [3], DSO [5], and ORB-SLAM [4]. It is obvious that any attempts to give a full relevant review would be incomplete. In this section, however, we skip the discussion on vision-only approaches, and only focus on the most relevant results on monocular visual-inertial state estimation.

The simplest way to deal with visual and inertial measurements is loosely-coupled sensor fusion [11], [12], where IMU is treated as an independent module to assist vision-only pose estimates obtained from the visual structure from motion. Fusion is usually done by an extended Kalman filter (EKF), where IMU is used for state propagation and the vision-only pose is used for the update. Further on, tightly-coupled visual-inertial algorithms are either based on the EKF [13]–[15] or graph optimization [7], [8], [16], [17], where camera and IMU measurements are jointly optimized from the raw measurement level. A popular EKF based VIO approach is MSCKF [13], [14]. MSCKF maintains several previous camera poses in the state vector, and uses visual measurements of the same feature across multiple camera views to form multi-constraint update. SR-ISWF [18], [19] is an extension of MSCKF. It uses square-root form [20] to achieve single-precision representation and avoid poor numerical properties. This approach employs the inverse filter for iterative re-linearization, making it equal to optimization-based algorithms. Batch graph optimization or bundle adjustment techniques maintain and optimize all measurements to obtain the optimal state estimates. To achieve constant processing time, popular graph-based VIO methods [8], [16], [17] usually optimize over a bounded-size sliding window of recent states by marginalizing out past states and measurements. Due to high computational demands of iterative solving of nonlinear systems, few graph-based can achieve real-time performance on resource-constrained platforms, such as mobile phones.

For visual measurement processing, algorithms can be categorized into either direct or indirect method according to the definition of visual residual models. Direct approaches [2], [3], [21] minimize photometric error while indirect approaches [8], [14], [16] minimize geometric displacement. Direct methods require a good initial guess due to their small region of attraction, while indirect approaches consume extra computational resources on extracting and matching features. Indirect approaches are more frequently found in real-world engineering deployment due to its maturity and robustness. However, direct approaches are easier to be extended for dense mapping as they are operated directly on the pixel level.

In practice, IMUs usually acquire data at a much higher rate than the camera. Different methods have been proposed to handle the high rate IMU measurements. The most straightforward approach is to use the IMU for state propagation in EKF-based approaches [11], [13]. In a graph optimization formulation, an efficient technique called IMU pre-integration is developed in

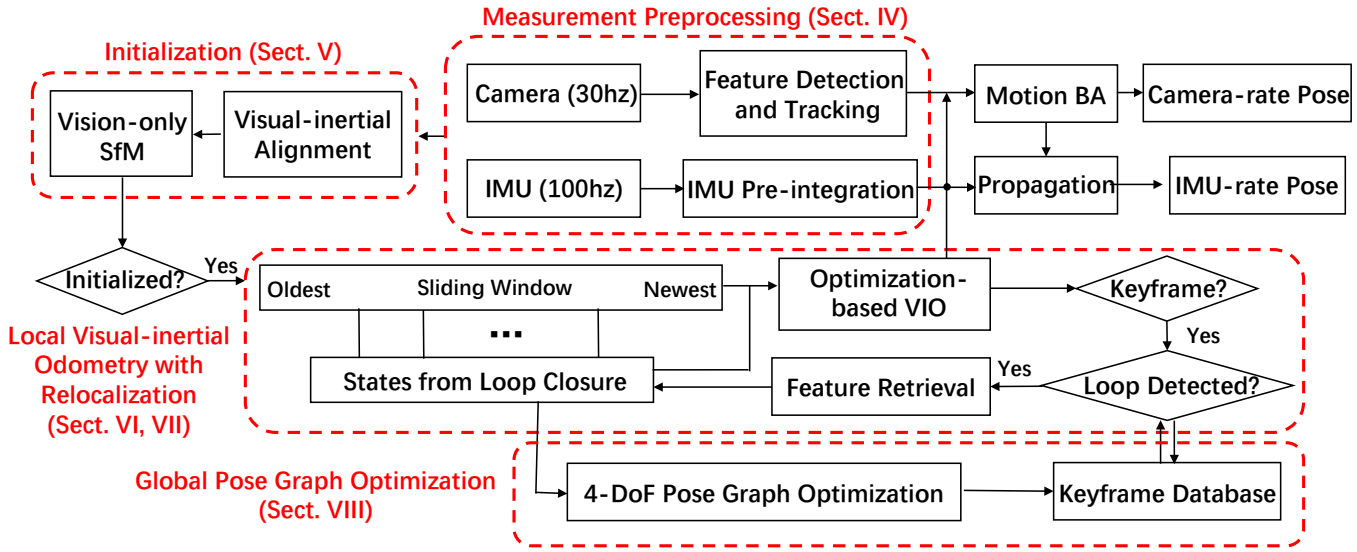


Fig. 2. A block diagram illustrating the full pipeline of the proposed monocular visual-inertial state estimator.

order to avoid repeated IMU re-integration This technique was first introduced in [22], which parametrize rotation error using Euler angles. An on-manifold rotation formulation for IMU-preintegration was developed in our previous work [7]. This work derived the covariance propagation using continuous-time IMU error state dynamics. However, IMU biases were ignored. The pre-integration theory was further improved in [23] by adding posterior IMU bias correction.

Accurate initial values are crucial to bootstrap any monocular VINS. A linear estimator initialization method that leverages relative rotations from short-term IMU pre-integration was proposed in [8], [24]. However, this method does not model gyroscope bias, and fails to model sensor noise in raw projection equations. In real-world applications, this results in unreliable initialization when visual features are far away from the sensor suite. A closed-form solution to the monocular visual-inertial initialization problem was introduced in [25]. Later, an extension to this closed-form solution by adding gyroscope bias calibration was proposed in [26]. These approaches fail to model the uncertainty in inertial integration since they rely on the double integration of IMU measurements over an extended period of time. In [27], a re-initialization and failure recovery algorithm based on SVO [2] was proposed. This is a practical method based on a loosely-coupled fusion framework. However, an additional downward-facing distance sensor is required to recover the metric scale. An initialization algorithm built on top of the popular ORB-SLAM [4] was introduced in [17]. An initial estimation of the scale, gravity direction, velocity and IMU biases are computed for the visual-inertial full BA given a set of keyframes from ORB-SLAM. However, it is reported that the time required for scale convergence can be longer than 10 seconds. This can pose problems for robotic navigation tasks that require scale estimates right at the beginning.

VIO approaches, regardless the underlying mathematical formulation that they rely on, suffer from long term drifting in global translation and orientation. To this end, loop closure plays an important role for long-term operations. ORB-SLAM [4] is able to close loops and reuse the map, which takes advantage of Bag-of-World [6]. A 7 DOF [28] (position, orientation, and scale) pose graph optimization is followed loop detection. In contrast, for monocular VINS, thanks to the addition of IMU, drift only occurs in 4 DOF, which is the 3D translation, and the rotation around the gravity direction (yaw angle). Therefore, in this paper, we choose to optimize the pose graph with loop constraints in the minimum 4 DOF setting.

III. Overview

The structure of proposed monocular visual-inertial state estimator is shown in Fig. 2. The system starts with measurement preprocessing (Sect. IV), in which features are extracted and tracked, and IMU measurements between two consecutive frames are pre-integrated. The initialization procedure (Sect. V provides all necessary values, including pose, velocity, gravity vector, gyroscope bias, and 3D feature location, for bootstrapping the subsequent nonlinear optimization-based VIO. The VIO (Sect. VI) with relocalization (Sect. VII) modules tightly fuses pre-integrated IMU measurements, feature observations, and re-detected features from loop closure. Finally, the pose graph optimization module (Sect. VIII) takes in geometrically verified relocalization results, and perform global optimization to eliminate drift. The VIO, relocalization, and pose graph optimization modules run concurrently in a multi-thread setting. Each module has different running rates and real-time guarantees to ensure reliable operation at all times.

We now define notations and frame definitions that we use throughout the paper. We consider $(\cdot)^w$ as the world frame. The direction of the gravity is aligned with the z axis of the world frame. $(\cdot)^b$ is the body frame, which we define it to be the same as the IMU frame. $(\cdot)^c$ is the camera frame. We use both rotation matrices $\mathbf{R}$ and Hamilton quaternions

$\mathbf{q}$ to represent rotation. We primarily use quaternions in state vectors, but rotation matrices are also used for convenience rotation of 3D vectors. $\mathbf{q}_b^w, \mathbf{p}_b^w$ are rotation and translation from the body frame to the world frame. b_k is the body frame while taking the k^{th} image. c_k is the camera frame while taking the k^{th} image. $\otimes$ represents the multiplication operation between two quaternions. $\mathbf{g}^w = [0, 0, g]^T$ is the gravity vector in the world frame. Finally, we denote $\hat{(\cdot)}$ as the noisy measurement or estimate of a certain quantity.

IV. Measurement Preprocessing

This section presents preprocessing steps for both inertial and monocular visual measurements. For visual measurements, we track features between consecutive frames and detect new features in the latest frame. For IMU measurements, we pre-integrate them between two consecutive frames. Note that the measurements of the low-cost IMU that we use are affected by both bias and noise. We therefore especially take bias into account in the IMU pre-integration process.

A. Vision Processing Front-end

For each new image, existing features are tracked by the KLT sparse optical flow algorithm [29]. Meanwhile, new corner features are detected [30] to maintain a minimum number (100-300) of features in each image. The detector enforces a uniform feature distribution by setting a minimum separation of pixels between two neighboring features. 2D Features are firstly undistorted and then projected to a unit sphere after passing outlier rejection. Outlier rejection is performed using RANSAC with fundamental matrix model [31].

Keyframes are also selected in this step. We have two criteria for keyframe selection. The first one is the average parallax apart from the previous keyframe. If the average parallax of tracked features is between the current frame and the latest keyframe is beyond a certain threshold, we treat frame as a new keyframe. Note that not only translation but also rotation can cause parallax. However, features cannot be triangulated in the rotation-only motion. To avoid this situation, we use short-term integration of gyroscope measurements to compensate rotation when calculating parallax. Note that this rotation compensation is only used to keyframe selection, and is not involved in rotation calculation in the VINS formulation. To this end, even if the gyroscope contains large noise or is biased, it will only result in suboptimal keyframe selection results, and will not directly affect the estimation quality. Another criterion is tracking quality. If the number of tracked features goes below a certain threshold, we treat this frame as a new keyframe. This criterion is to avoid complete loss of feature tracks.

B. IMU Pre-integration

IMU Pre-integration was first proposed in [22], which parametrized rotation error in Euler angle. An on-manifold rotation formulation for IMU pre-integration was developed in our previous work [7]. This work derived the covariance propagation using continuous-time IMU error state dynamics. However, IMU biases were ignored. The pre-integration theory was further improved in [23] by adding posterior IMU bias correction. In this paper, we extend the IMU pre-integration proposed in our previous work [7] by incorporating IMU bias correction.

The raw gyroscope and accelerometer measurements from IMU, $\hat{\boldsymbol{\omega}}$ and $\hat{\mathbf{a}}$, are given by:

$$\begin{aligned} \hat{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_{a_t} + \mathbf{R}_w^t \mathbf{g}^w + \mathbf{n}_a \\ \hat{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_{w_t} + \mathbf{n}_w. \end{aligned} \tag{1}$$

IMU measurements, which are measured in the body frame, combines the force for countering gravity and the platform dynamics, and are affected by acceleration bias $\mathbf{b}_a$, gyroscope bias $\mathbf{b}_w$, and additive noise. We assume that the additive noise in acceleration and gyroscope measurements are Gaussian, $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_a^2)$, $\mathbf{n}_w \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_w^2)$. Acceleration bias and gyroscope bias are modeled as random walk, whose derivatives are Gaussian, $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_a}^2)$, $\mathbf{n}_{b_w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_{b_w}^2)$:

$$\dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \quad \dot{\mathbf{b}}_{w_t} = \mathbf{n}_{b_w}. \tag{2}$$

Given two time instants that correspond to image frames b_k and b_{k+1}, position, velocity, and orientation states can be propagated by inertial measurements during time interval $[t_k, t_{k+1}]$ in the world frame:

$$\begin{aligned} \mathbf{p}_{b_{k+1}}^w &= \mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k \\ &\quad + \iint_{t\in[t_k,t_{k+1}]} (\mathbf{R}_t^w(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}^w)\, dt^2 \\ \mathbf{v}_{b_{k+1}}^w &= \mathbf{v}_{b_k}^w + \int_{t\in[t_k,t_{k+1}]} (\mathbf{R}_t^w(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a) - \mathbf{g}^w)\, dt \\ \mathbf{q}_{b_{k+1}}^w &= \mathbf{q}_{b_k}^w \otimes \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t} - \mathbf{n}_w)\mathbf{q}_t^{b_k} dt, \end{aligned} \tag{3}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{4}$$

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

It can be seen that the IMU state propagation requires rotation, position and velocity of frame b_k. When these starting states change, we need to re-propagate IMU measurements. Especially in the optimization-based algorithm, every time we adjust poses, we will need to re-propagate IMU measurements between them. This propagation strategy is computationally demanding. To avoid re-propagation, we adopt pre-integration algorithm.

After change the reference frame from the world frame to the local frame b_k, we can only pre-integrate the parts which are related to linear acceleration $\hat{\mathbf{a}}$ and angular velocity $\hat{\boldsymbol{\omega}}$ as follows:

$$\begin{aligned} \mathbf{R}_w^{b_k} \mathbf{p}_{b_{k+1}}^w &= \mathbf{R}_w^{b_k}(\mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k - \frac{1}{2}\mathbf{g}^w \Delta t_k^2) + \boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\ \mathbf{R}_w^{b_k} \mathbf{v}_{b_{k+1}}^w &= \mathbf{R}_w^{b_k}(\mathbf{v}_{b_k}^w - \mathbf{g}^w \Delta t_k) + \boldsymbol{\beta}_{b_{k+1}}^{b_k} \\ \mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^w &= \boldsymbol{\gamma}_{b_{k+1}}^{b_k}, \end{aligned} \tag{5}$$

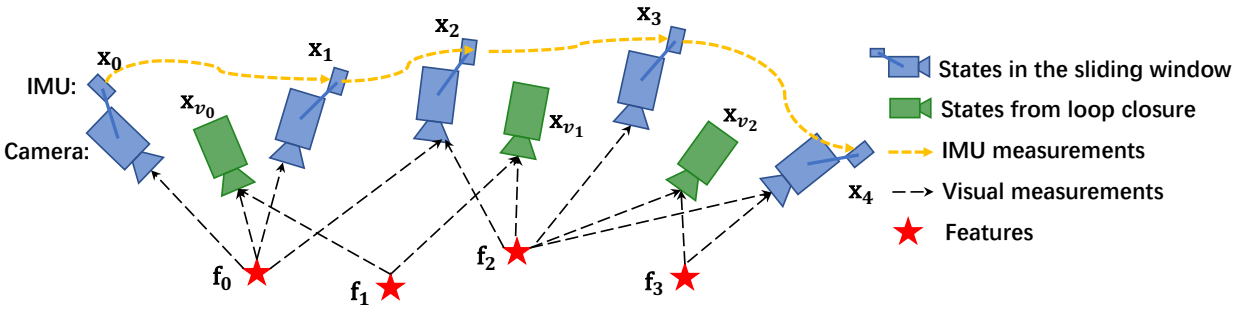


Fig. 3. An illustration of the sliding window monocular VIO with relocalization. It is a tightly-coupled formulation with IMU, visual, and loop measurements.

where,

$$\begin{aligned}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \iint_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a)dt^2 \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a)dt \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t} - \mathbf{n}_w)\boldsymbol{\gamma}_t^{b_k}dt.\end{aligned} \tag{6}$$

It can be seen that the pre-integration terms (6) can be obtained solely with IMU measurements by taking b_k as the reference frame. $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}, \boldsymbol{\beta}_{b_{k+1}}^{b_k}, \boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ are only related to IMU biases instead of other states in b_k and b_{k+1}. When the estimation of bias changes, if the change is small, we adjust $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}, \boldsymbol{\beta}_{b_{k+1}}^{b_k}, \boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ by their first-order approximations with respect to the bias, otherwise we do re-propagation. This strategy saves a significant amount of computational resources for optimization-based algorithms, since we do not need to propagate IMU measurements repeatedly.

For discrete-time implementation, different numerical integration methods such as Euler, mid-point, RK4 integration can be applied. Here Euler integration is chosen to demonstrate the procedure for easy understanding (we use mid-point integration in the implementation code).

At the beginning, $\boldsymbol{\alpha}_{b_k}^{b_k}, \boldsymbol{\beta}_{b_k}^{b_k}$ is $\mathbf{0}$, and $\boldsymbol{\gamma}_{b_k}^{b_k}$ is identity quaternion. The mean of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ in (6) is propagated step by step as follows. Note that the additive noise terms $\mathbf{n}_a$, $\mathbf{n}_w$ are unknown, and is treated as zero in the implementation. This results in estimated values of the pre-integration terms, marked by $\hat{(\cdot)}$:

$$\begin{aligned}\hat{\boldsymbol{\alpha}}_{i+1}^{b_k} &= \hat{\boldsymbol{\alpha}}_i^{b_k} + \hat{\boldsymbol{\beta}}_i^{b_k}\delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{\gamma}}_i^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t^2 \\ \hat{\boldsymbol{\beta}}_{i+1}^{b_k} &= \hat{\boldsymbol{\beta}}_i^{b_k} + \mathbf{R}(\hat{\boldsymbol{\gamma}}_i^{b_k})(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t \\ \hat{\boldsymbol{\gamma}}_{i+1}^{b_k} &= \hat{\boldsymbol{\gamma}}_i^{b_k} \otimes \begin{bmatrix}1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{w_i})\delta t\end{bmatrix}\end{aligned} . \tag{7}$$

i is discrete moment corresponding to a IMU measurement within $[t_k, t_{k+1}]$. δt is the time interval between two IMU measurements i and $i+1$.

Then we deal with the covariance propagation. Since the four-dimensional rotation quaternion $\boldsymbol{\gamma}_t^{b_k}$ is over-parameterized, we define its error term as a perturbation around its mean:

$$\boldsymbol{\gamma}_t^{b_k} \approx \hat{\boldsymbol{\gamma}}_t^{b_k} \otimes \begin{bmatrix}1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_t^{b_k}\end{bmatrix}, \tag{8}$$

where $\delta\boldsymbol{\theta}_t^{b_k}$ is three-dimensional small perturbation.

We can derive continuous-time linearized dynamics of error terms of (6):

$$\begin{aligned}\begin{bmatrix}\delta\dot{\boldsymbol{\alpha}}_t^{b_k} \\ \delta\dot{\boldsymbol{\beta}}_t^{b_k} \\ \delta\dot{\boldsymbol{\theta}}_t^{b_k} \\ \delta\dot{\mathbf{b}}_{a_t} \\ \delta\dot{\mathbf{b}}_{w_t}\end{bmatrix} &= \begin{bmatrix}0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{R}_t^{b_k}\lfloor\hat{\mathbf{a}}_t - \mathbf{b}_{a_t}\rfloor_\times & -\mathbf{R}_t^{b_k} & 0 \\ 0 & 0 & -\lfloor\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t}\rfloor_\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\alpha}_t^{b_k} \\ \delta\boldsymbol{\beta}_t^{b_k} \\ \delta\boldsymbol{\theta}_t^{b_k} \\ \delta\mathbf{b}_{a_t} \\ \delta\mathbf{b}_{w_t}\end{bmatrix} \\ &+ \begin{bmatrix}0 & 0 & 0 & 0 \\ -\mathbf{R}_t^{b_k} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{n}_a \\ \mathbf{n}_w \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_w}\end{bmatrix} = \mathbf{F}_t\delta\mathbf{z}_t^{b_k} + \mathbf{G}_t\mathbf{n}_t,\end{aligned} \tag{9}$$

$\mathbf{P}_{b_{k+1}}^{b_k}$ can be computed recursively by first-order discrete-time covariance update with the initial covariance $\mathbf{P}_{b_k}^{b_k} = \mathbf{0}$:

$$\begin{aligned}\mathbf{P}_{t+\delta t}^{b_k} = (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{P}_t^{b_k}(\mathbf{I} + \mathbf{F}_t\delta t)^T + (\mathbf{G}_t\delta t)\mathbf{Q}(\mathbf{G}_t\delta t)^T, \\ t \in [k, k+1],\end{aligned} \tag{10}$$

where $\mathbf{Q}$ is the diagonal covariance matrix of noise $(\boldsymbol{\sigma}_a^2, \boldsymbol{\sigma}_w^2, \boldsymbol{\sigma}_{b_a}^2, \boldsymbol{\sigma}_{b_w}^2)$.

Meanwhile, the first-order Jacobian matrix $\mathbf{J}_{b_{k+1}}$ of $\delta\mathbf{z}_{b_{k+1}}^{b_k}$ with respect to $\delta\mathbf{z}_{b_k}^{b_k}$ can be also compute recursively with the initial Jacobian $\mathbf{J}_{b_k} = \mathbf{I}$,

$$\mathbf{J}_{t+\delta t} = (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{J}_t, \quad t \in [k, k+1]. \tag{11}$$

Using this recursive formulation, we get the covariance matrix $\mathbf{P}_{b_{k+1}}^{b_k}$ and the Jacobian $\mathbf{J}_{b_{k+1}}$. The first order approximation 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 biases can be write as:

$$\begin{aligned}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\alpha}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\alpha}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\beta}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\beta}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix}1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\delta\mathbf{b}_{w_k}\end{bmatrix}\end{aligned} \tag{12}$$

where $\mathbf{J}_{b_a}^{\alpha}$ and is the sub-block matrix in $\mathbf{J}_{b_{k+1}}$ whose location is corresponding to $\frac{\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{a_k}}$. The same meaning is also used for $\mathbf{J}_{b_w}^{\alpha}, \mathbf{J}_{b_a}^{\beta}, \mathbf{J}_{b_w}^{\beta}, \mathbf{J}_{b_w}^{\gamma}$. When the estimation of bias changes

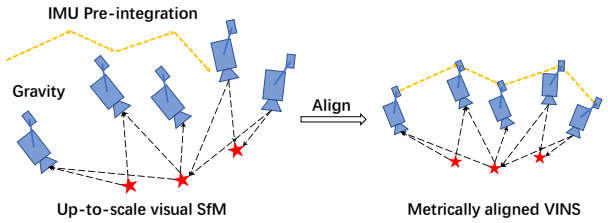


Fig. 4. An illustration of the visual-inertial alignment process for estimator initialization.

slightly, we use (12) to correct pre-integration results approximately instead of re-propagation.

Now we are able to write down the IMU measurement model with its corresponding covariance $\mathbf{P}_{b_{k+1}}^{b_k}$:

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k\right) \\ \mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w\right) \\ \mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^w \\ \mathbf{b}_{a b_{k+1}} - \mathbf{b}_{a b_k} \\ \mathbf{b}_{w b_{k+1}} - \mathbf{b}_{w b_k} \end{bmatrix}. \tag{13}$$

V. Estimator Initialization

Monocular tightly-coupled visual-inertial odometry is a highly nonlinear system. Since the scale is not directly observable from a monocular camera, it is hard to directly fuse these two measurements without good initial values. One may assume a stationary initial condition to start the monocular VINS estimator. However, this assumption is inappropriate as initialization under motion is frequently encountered in real-world applications. The situation becomes even more complicated when IMU measurements are corrupted by large bias. In fact, initialization is usually the most fragile step for monocular VINS. A robust initialization procedure is needed to ensure the applicability of the system.

We adopt a loosely-coupled sensor fusion method to get initial values. We find that vision-only SLAM, or Structure from Motion (SfM), has a good property of initialization. In most cases, a visual-only system can bootstrap itself by derived initial values from relative motion methods, such as the eight-point [32] or five-point [33] algorithms, or estimating the homogeneous matrices. By aligning metric IMU pre-integration with the visual-only SfM results, we can roughly recover scale, gravity, velocity, and even bias. This is sufficient for bootstrapping a nonlinear monocular VINS estimator, as shown in Fig. 4.

In contrast to [17], which simultaneously estimates gyroscope and accelerometer bias during the initialization phase, we choose to ignore accelerometer bias terms in the initial step. Accelerometer bias is coupled with gravity, and due to the large magnitude of the gravity vector comparing to platform dynamics, and the relatively short during of the initialization phase, these bias terms are hard to be observed. A detailed analysis of accelerometer bias calibration is presented in our previous work [34].

A. Sliding Window Vision-Only SfM

The initialization procedure starts with a vision-only SfM to estimate a graph of up-to-scale camera poses and feature positions.

We maintain a sliding window of frames for bounded computational complexity. Firstly, we check feature correspondences between the latest frame and all previous frames. If we can find stable feature tracking (more than 30 tracked features) and sufficient parallax (more than 20 rotation-compensated pixels) between the latest frame and any other frames in the sliding window. we recover the relative rotation and up-to-scale translation between these two frames using the Five-point algorithm [33]. Otherwise, we keep the latest frame in the window and wait for new frames. If the five-point algorithm success, we arbitrarily set the scale and triangulate all features observed in these two frames. Based on these triangulated features, a perspective-n-point (PnP) method [35] is performed to estimate poses of all other frames in the window. Finally, a global full bundle adjustment [36] is applied to minimize the total reprojection error of all feature observations. Since we do not yet have any knowledge about the world frame, we set the first camera frame $(\cdot)^{c_0}$ as the reference frame for SfM. All frame poses $(\bar{\mathbf{p}}_{c_k}^{c_0}, \mathbf{q}_{c_k}^{c_0})$ and feature positions are represented with respect to $(\cdot)^{c_0}$. Suppose we have a rough measure extrinsic parameters $(\mathbf{p}_c^b, \mathbf{q}_c^b)$ between the camera and the IMU, we can translate poses from camera frame to body (IMU) frame,

$$\begin{aligned} \mathbf{q}_{b_k}^{c_0} &= \mathbf{q}_{c_k}^{c_0} \otimes \left(\mathbf{q}_c^b\right)^{-1} \\ s\bar{\mathbf{p}}_{b_k}^{c_0} &= s\bar{\mathbf{p}}_{c_k}^{c_0} - \mathbf{R}_{b_k}^{c_0}\mathbf{p}_c^b, \end{aligned} \tag{14}$$

where s the scaling parameter that aligns the visual structure to the metric scale. Solving this scaling parameter is the key to achieve successful initialization.

B. Visual-Inertial Alignment

1) Gyroscope Bias Calibration: Consider two consecutive frames b_k and b_{k+1} in the window, we get the rotation $\mathbf{q}_{b_k}^{c_0}$ and $\mathbf{q}_{b_{k+1}}^{c_0}$ from the visual SfM, as well as the relative constraint $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$ from IMU pre-integration. We linearize the IMU pre-integration term with respect to gyroscope bias and minimize the following cost function:

$$\begin{aligned} &\min_{\delta b_w} \sum_{k\in\mathcal{B}} \left\| \mathbf{q}_{b_{k+1}}^{c_0}{}^{-1} \otimes \mathbf{q}_{b_k}^{c_0} \otimes \boldsymbol{\gamma}_{b_{k+1}}^{b_k} \right\|^2 \\ &\boldsymbol{\gamma}_{b_{k+1}}^{b_k} \approx \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\delta\mathbf{b}_w \end{bmatrix}, \end{aligned} \tag{15}$$

where $\mathcal{B}$ indexes all frames in the window. We have the first order approximation of $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$ with respect to the gyroscope bias using the bias Jacobian derived in Sect. IV-B. In such way, we get an initial calibration of the gyroscope bias $\mathbf{b}_w$. Then we re-propagate all IMU pre-integration terms $\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k}$, $\hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}$, and $\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}$ using the new gyroscope bias.

2) Velocity, Gravity Vector and Metric Scale Initialization: After the gyroscope bias is initialized, we move on to initialize

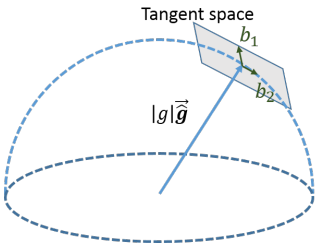


Fig. 5. Illustration of 2 DOF parameterization of gravity. Since the magnitude of gravity is known, $\mathbf{g}$ lies on a sphere with radius $g \approx 9.81m/s^2$. The gravity is parameterized around current estimate as $g \cdot \bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, where $\mathbf{b}_1$ and $\mathbf{b}_2$ are two orthogonal basis spanning the tangent space.

other essential states for navigation, namely velocity, gravity vector, and metric scale:

$$\mathcal{X}_I = \left[\mathbf{v}_{b_0}^{b_0},\, \mathbf{v}_{b_1}^{b_1},\, \cdots \mathbf{v}_{b_n}^{b_n},\, \mathbf{g}^{c_0},\, s\right], \tag{16}$$

where $\mathbf{v}_{b_k}^{b_k}$ is velocity in the body frame while taking the k^{th} image, $\mathbf{g}^{c_0}$ is the gravity vector in the c_0 frame, and s scales the monocular SfM to metric units.

Consider two consecutive frames b_k and b_{k+1} in the window, then (5) can be written as:

$$\begin{aligned}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \mathbf{R}_{c_0}^{b_k}(s(\bar{\mathbf{p}}_{b_{k+1}}^{c_0} - \bar{\mathbf{p}}_{b_k}^{c_0}) + \frac{1}{2}\mathbf{g}^{c_0}\Delta t_k^2 - \mathbf{R}_{b_k}^{c_0}\mathbf{v}_{b_k}^{b_k}\Delta t_k)\\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \mathbf{R}_{c_0}^{b_k}(\mathbf{R}_{b_{k+1}}^{c_0}\mathbf{v}_{b_{k+1}}^{b_{k+1}} + \mathbf{g}^{c_0}\Delta t_k - \mathbf{R}_{b_k}^{c_0}\mathbf{v}_{b_k}^{b_k}).\end{aligned} \tag{17}$$

We can combine (14) and (17) into the following linear measurement model:

$$\hat{\mathbf{z}}_{b_{k+1}}^{b_k} = \begin{bmatrix}\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} - \mathbf{p}_c^b + \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0}\mathbf{p}_c^b \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k}\end{bmatrix} = \mathbf{H}_{b_{k+1}}^{b_k}\mathcal{X}_I + \mathbf{n}_{b_{k+1}}^{b_k} \tag{18}$$

where,

$$\mathbf{H}_{b_{k+1}}^{b_k} = \begin{bmatrix}-\mathbf{I}\Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}_{c_0}^{b_k}\Delta t_k^2 & \mathbf{R}_{c_0}^{b_k}(\bar{\mathbf{p}}_{c_{k+1}}^{c_0} - \bar{\mathbf{p}}_{c_k}^{c_0}) \\ -\mathbf{I} & \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0} & \mathbf{R}_{c_0}^{b_k}\Delta t_k & \mathbf{0}\end{bmatrix} \tag{19}$$

It can be seen that $\mathbf{R}_{b_k}^{c_0}, \mathbf{R}_{b_{k+1}}^{c_0}, \bar{\mathbf{p}}_{c_k}^{c_0}, \bar{\mathbf{p}}_{c_{k+1}}^{c_0}$ are obtained from the up-to-scale monocular visual SfM. Δt_k is the time interval between two consecutive frames. By solving this linear least square problem:

$$\min_{\mathcal{X}_I} \sum_{k\in\mathcal{B}} \left\| \hat{\mathbf{z}}_{b_{k+1}}^{b_k} - \mathbf{H}_{b_{k+1}}^{b_k}\mathcal{X}_I \right\|^2, \tag{20}$$

we can get body frame velocities for every frame in the window, the gravity vector in the visual reference frame $(\cdot)^{c_0}$, as well as the scale parameter.

3) Gravity Refinement: The gravity vector obtained from the previous linear initialization step can be refined by constraining the magnitude. In most cases, the magnitude of gravity vector is known. This results in only 2 DOF remaining for the gravity vector. We therefore re-parameterize the gravity with two variables on its tangent space. Our parameterization represents the gravity vector as $g \cdot \bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, where g is the know magnitude of the gravity, $\bar{\hat{\mathbf{g}}}$ is a unit vector representing the gravity direction. $\mathbf{b}_1$ and $\mathbf{b}_2$ are two orthogonal basis spanning the tangent plane, as shown in Fig. 5. w_1 and w_2 are corresponding displacements towards $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. We can find one set of $\mathbf{b}_1$, $\mathbf{b}_2$ by cross products operations using Algorithm 1. Then we substitute $\mathbf{g}$ in (17) by $g \cdot \bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, and solve for w_1 and w_2 together with other state variables. This process iterates until $\hat{\mathbf{g}}$ converges.

Algorithm 1: Finding $\mathbf{b}_1$ and $\mathbf{b}_2$

```
if ĝ̄ ≠ [1, 0, 0] then
    b1 ← normalize(ĝ̄ × [1, 0, 0]);
else
    b1 ← normalize(ĝ̄ × [0, 0, 1]);
end
b2 ← ĝ̄ × b1;
```

4) Completing Initialization: After refining the gravity vector, we can get the rotation $\mathbf{q}_{c_0}^w$ between the world frame and the camera frame c_0 by rotating the gravity to the z-axis. We then rotate all variables from reference frame $(\cdot)^{c_0}$ to the world frame $(\cdot)^w$. The body frame velocities will also be rotated to world frame. Translational components from the visual SfM will be scaled to metric units. At this point, the initialization procedure is completed and all these metric values will be fed for a tightly-coupled monocular VIO.

VI. Tightly-coupled Monocular VIO

After estimator initialization, we proceed with a sliding window-based tightly-coupled monocular VIO for high-accuracy and robust state estimation. An illustration of the sliding window formulation is shown in Fig. 3.

A. Formulation

The full state vector in the sliding window is defined as:

$$\begin{aligned}\mathcal{X} &= \left[\mathbf{x}_0,\, \mathbf{x}_1,\, \cdots \mathbf{x}_n,\, \mathbf{x}_c^b,\, \lambda_0,\, \lambda_1,\, \cdots \lambda_m\right]\\ \mathbf{x}_k &= \left[\mathbf{p}_{b_k}^w,\, \mathbf{v}_{b_k}^w,\, \mathbf{q}_{b_k}^w,\, \mathbf{b}_a,\, \mathbf{b}_g\right], k \in [0, n]\\ \mathbf{x}_c^b &= \left[\mathbf{p}_c^b,\, \mathbf{q}_c^b\right],\end{aligned} \tag{21}$$

where $\mathbf{x}_k$ is the IMU state at the time that the k^{th} image is captured. It contains position, velocity, and orientation of the IMU in the world frame, and acceleration bias and gyroscope bias in the IMU body frame. n is the total number of keyframes, and m is the total number of features in the sliding window. λ_l is the inverse depth of the l^{th} feature from its first observation.

We use a visual-inertial bundle adjustment formulation. We minimize the sum of prior and the Mahalanobis norm of all measurement residuals to obtain a maximum posteriori estimation:

$$\begin{aligned}\min_{\mathcal{X}} \Bigg\{ &\left\|\mathbf{r}_p - \mathbf{H}_p\mathcal{X}\right\|^2 + \sum_{k\in\mathcal{B}} \left\|\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X})\right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \\ &\sum_{(l,j)\in\mathcal{C}} \rho(\left\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X})\right\|_{\mathbf{P}_l^{c_j}}^2)\Bigg\},\end{aligned} \tag{22}$$

where the Huber norm [37] is defined as:

$$\rho(s) = \begin{cases}1 & s \geq 1,\\ 2\sqrt{s} - 1 & s < 1.\end{cases} \tag{23}$$

$\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X})$ and $\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_{l}, \mathcal{X})$ are residuals for IMU and visual measurements respectively. Detailed definition of the residual terms will be presented in Sect. VI-B and Sect. VI-C. $\mathcal{B}$ is the set of all IMU measurements, $\mathcal{C}$ is the set of features which have been observed at least twice in the current sliding window. $\{\mathbf{r}_p, \mathbf{H}_p\}$ is the prior information from marginalization. Ceres Solver [38] is used for solving this nonlinear problem.

B. IMU Measurement Residual

Consider the IMU measurements within two consecutive frames b_k and b_{k+1} in the sliding window, according to the IMU measurement model defined in (13), the residual for pre-integrated IMU measurement can be defined as:

$$\begin{aligned}\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) &= \begin{bmatrix}\delta\boldsymbol{\alpha}^{b_k}_{b_{k+1}}\\ \delta\boldsymbol{\beta}^{b_k}_{b_{k+1}}\\ \delta\boldsymbol{\theta}^{b_k}_{b_{k+1}}\\ \delta\mathbf{b}_a\\ \delta\mathbf{b}_g\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{R}^{b_k}_{w}(\mathbf{p}^{w}_{b_{k+1}} - \mathbf{p}^{w}_{b_k} + \frac{1}{2}\mathbf{g}^{w}\Delta t_k^2 - \mathbf{v}^{w}_{b_k}\Delta t_k) - \hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}}\\ \mathbf{R}^{b_k}_{w}(\mathbf{v}^{w}_{b_{k+1}} + \mathbf{g}^{w}\Delta t_k - \mathbf{v}^{w}_{b_k}) - \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}}\\ 2\left[\mathbf{q}^{w^{-1}}_{b_k}\otimes\mathbf{q}^{w}_{b_{k+1}}\otimes(\hat{\boldsymbol{\gamma}}^{b_k}_{b_{k+1}})^{-1}\right]_{xyz}\\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k}\\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k}\end{bmatrix},\end{aligned} \tag{24}$$

where $\left[\cdot\right]_{xyz}$ extracts the vector part of a quaternion $\mathbf{q}$ for error state representation. $\delta\boldsymbol{\theta}^{b_k}_{b_{k+1}}$ is the three dimensional error-state representation of quaternion. $[\hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}}, \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}}, \hat{\boldsymbol{\gamma}}^{b_k}_{b_{k+1}}]^{\mathrm{T}}$ are pre-integrated IMU measurement terms using only noisy accelerometer and gyroscope measurements within the time interval between two consecutive image frames. Accelerometer and gyroscope biases are also included in the residual terms for online correction.

C. Visual Measurement Residual

In contrast to traditional pinhole camera models that define reprojection errors on a generalized image plane, we define the camera measurement residual on a unit sphere. The optics for almost all types of cameras, including wide-angle, fisheye or omnidirectional cameras, can be modeled as a unit ray connecting the surface of a unit sphere. Consider the l^{th} feature that is first observed in the i^{th} image, the residual for the feature observation in the j^{th} image is defined as:

$$\begin{aligned}\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_{l}, \mathcal{X}) &= \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}^T\cdot(\hat{\mathcal{P}}^{c_j}_{l} - \frac{\mathcal{P}^{c_j}_{l}}{\|\mathcal{P}^{c_j}_{l}\|})\\ \hat{\mathcal{P}}^{c_j}_{l} &= {\pi_c}^{-1}(\begin{bmatrix}\hat{u}^{c_j}_{l}\\ \hat{v}^{c_j}_{l}\end{bmatrix})\\ \mathcal{P}^{c_j}_{l} &= \mathbf{R}^{c}_{b}(\mathbf{R}^{b_j}_{w}(\mathbf{R}^{w}_{b_i}(\mathbf{R}^{b}_{c}\frac{1}{\lambda_l}{\pi_c}^{-1}(\begin{bmatrix}u^{c_i}_{l}\\ v^{c_i}_{l}\end{bmatrix})\\ &\qquad + \mathbf{p}^{b}_{c}) + \mathbf{p}^{w}_{b_i} - \mathbf{p}^{w}_{b_j}) - \mathbf{p}^{b}_{c}),\end{aligned} \tag{25}$$

where $[u^{c_i}_l, v^{c_i}_l]$ is the first observation of the l^{th} feature that happens in the i^{th} image. $[\hat{u}^{c_j}_l, \hat{v}^{c_j}_l]$ is the observation of the same feature in the j^{th} image. π_c^{-1} is the back projection function which turns a pixel location into a unit vector using camera intrinsic parameters. Since the degrees-of-freedom of the vision residual is two, we project the residual vector onto the tangent plane. $\mathbf{b}_1, \mathbf{b}_2$ are two arbitrarily selected orthogonal bases that span the tangent plane of $\hat{\mathcal{P}}^{c_j}_{l}$, as shown in Fig. 6. We can find one set of $\mathbf{b}_1, \mathbf{b}_2$ easily, as shown in Algorithm 1. $\mathbf{P}^{c_j}_{l}$, as used in (22), is the standard covariance of a fixed length in the tangent space.

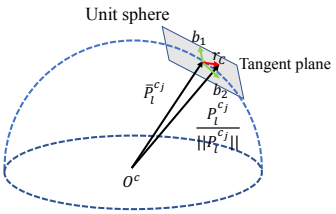


Fig. 6. An illustration of the visual residual on a unit sphere. $\hat{\bar{\mathcal{P}}}^{c_j}_{l}$ is the unit vector for the observation of the l^{th} feature in the j^{th} frame. $\mathcal{P}^{c_j}_{l}$ is predicted feature measurement on the unit sphere by transforming its first observation in the i^{th} frame to the j^{th} frame. The residual is defined on the tangent plane of $\hat{\bar{\mathcal{P}}}^{c_j}_{l}$.

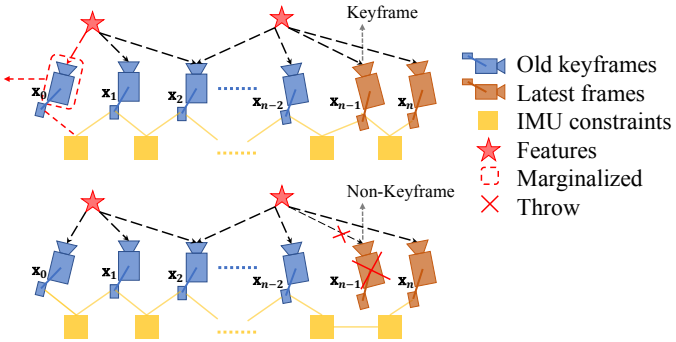


Fig. 7. An illustration of our marginalization strategy. If the second latest frame is a keyframe, we will keep it in the window, and marginalize the oldest frame and its corresponding visual and inertial measurements. Marginalized measurements are turned into a prior. If the second latest frame is not a keyframe, we will simply remove the frame and all its corresponding visual measurements. However, pre-integrated inertial measurements are kept for non-keyframes, and the pre-integration process is continued towards the next frame.

D. Marginalization

In order to bound the computational complexity of our optimization-based VIO, marginalization is incorporated. We selectively marginalize out IMU states $\mathbf{x}_k$ and features λ_l from the sliding window, meanwhile convert measurements corresponding to marginalized states into a prior.

As shown in the Fig. 7, when the second latest frame is a keyframe, it will stay in the window, and the oldest frame is marginalized out with its corresponding measurements. Otherwise, if the second latest frame is a non-keyframe, we throw visual measurements and keep IMU measurements that connect to this non-keyframe. We do not marginalize out all measurements for non-keyframes in order to maintain sparsity of the system. Our marginalization scheme aims to keep spatially separated keyframes in the window. This ensures

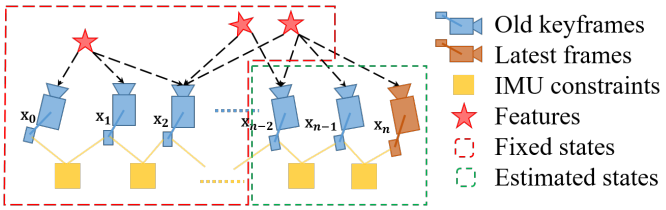


Fig. 8. An illustration of motion-only bundle adjustment for camera-rate outputs.

sufficient parallax for feature triangulation, and maximize the probability of maintaining accelerometer measurements with large excitation.

The marginalization is carried out using the Schur complement [39]. We construct a new prior based on all marginalized measurements related to the removed state. The new prior is added onto the existing prior.

We do note that marginalization results in the early fix of linearization points, which may result in suboptimal estimation results. However, since small drifting is acceptable for VIO, we argue that the negative impact caused by marginalization is not critical.

E. Motion-only Visual-Inertial Bundle Adjustment for Camera-Rate State Estimation

For devices with low computational power, such as mobile phones, the tightly-coupled monocular VIO cannot achieve camera-rate outputs due to the heavy computation demands for the nonlinear optimization. To this end, we employ a lightweight motion-only visual-inertial bundle adjustment to boost the state estimation to camera-rate (30 Hz).

The cost function for the motion-only visual-inertial bundle adjustment is the same as the one for monocular VIO in (22). However, instead of optimizing all states in the sliding window, we only optimize the poses and velocities of a fixed number of latest IMU states. We treat feature depth, extrinsic parameters, bias, and old IMU states that we do not want to optimize as constant values. We do use all visual and inertial measurements for the motion-only bundle adjustment. This results in much smoother state estimates than single frame PnP methods. An illustration of the proposed strategy is shown in Fig. 8. In contrast to the full tightly-coupled monocular VIO, which may cause more than 50ms on state-of-the-art embedded computers, the motion-only visual-inertial bundle adjustment only takes about 5ms to compute. This enables the low-latency camera-rate pose estimation that is particularly beneficial for drone and AR applications.

F. IMU Forward Propagation for IMU-Rate State Estimation

IMU measurements come at a much higher rate than visual measurements. Although the frequency of our VIO is limited by image capture frequency, we can still directly propagate the latest VIO estimate with the set of most recent IMU measurements to achieve IMU-rate performance. The high-frequency state estimates can be utilized as state feedback for closed loop closure. An autonomous flight experiment utilizing this IMU-rate state estimates is presented in Sect. IX-D.

G. Failure Detection and Recovery

Although our tightly-coupled monocular VIO is robust to various challenging environments and motions. Failure is still unavoidable due to violent illumination change or severely aggressive motions. Active failure detection and recovery strategy can improve the practicability of proposed system. Failure detection is an independent module that detects unusual output from the estimator. We are currently using the following criteria for failure detection:

- The number of features being tracking in the latest frame is less than a certain threshold;
- Large discontinuity in position or rotation between last two estimator outputs;
- Large change in bias or extrinsic parameters estimation;

Once a failure is detected, the system switches back to the initialization phase. Once the monocular VIO is successfully initialized, a new and separate segment of the pose graph will be created.

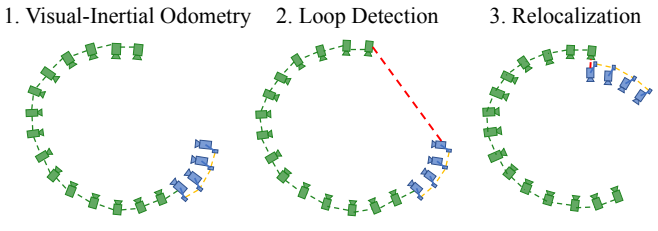
VII. Relocalization

Our sliding window and marginalization scheme bound the computation complexity, but it also introduces accumulated drifts for the system. To be more specific, drifts occur in global 3D position (x, y, z) and the rotation around the gravity direction (yaw). To eliminate drifts, a tightly-coupled relocalization module that seamlessly integrates with the monocular VIO is proposed. The relocalization process starts with a loop detection module that identifies places that have already been visited. Feature-level connections between loop closure candidates and the current frame are then established. These feature correspondences are tightly integrated into the monocular VIO module, resulting in drift-free state estimates with minimum computation overhead. Multiple observations of multiple features are directly used for relocalization, resulting in higher accuracy and better state estimation smoothness. A graphical illustration of the relocalization procedure is shown in Fig. 9(a).

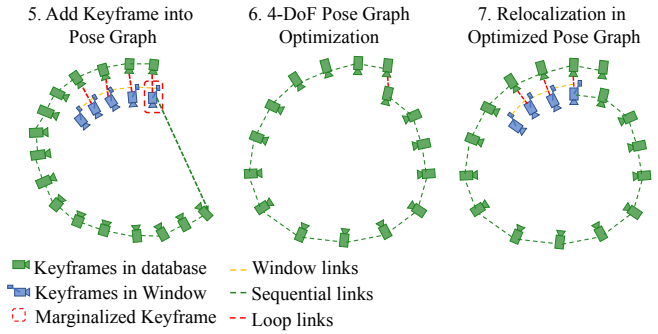
A. Loop Detection

We utilize DBoW2 [6], a state-of-the-art bag-of-word place recognition approach, for loop detection. In addition to the corner features that are used for the monocular VIO, 500 more corners are detected and described by the BRIEF descriptor [40]. The additional corner features are used to achieve better recall rate on loop detection. Descriptors are treated as the visual word to query the visual database. DBoW2 returns loop closure candidates after temporal and geometrical consistency check. We keep all BRIEF descriptors for feature retrieving, but discard the raw image to reduce memory consumption.

We note that our monocular VIO is able to render roll and pitch angles observable. As such, we do not need to rely on rotation-invariant features, such as the ORB feature used in ORB SLAM [4].



(a) Relocalization



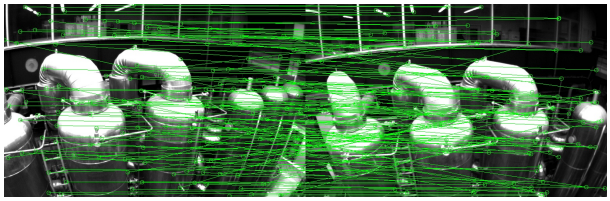
(b) Global Pose Graph optimization

Fig. 9. A diagram illustrating the relocalization and pose graph optimization procedure. Fig. 9(a) shows the relocalization procedure. It starts with VIO-only pose estimates (blue). Past states are recorded (green). If a loop is detected for the newest keyframe (Sect. VII-A), as shown by the red line in the second plot, a relocalization occurred. Note that due to the use of feature-level correspondences for relocalization, we are able to incorporate loop closure constraints from multiple past keyframes (Sect. VII-C), as indicated in the last three plots. The pose graph optimization is illustrated in Fig. 9(b). A keyframe is added into the pose graph when it is marginalized out from the sliding window. If there is a loop between this keyframe and any other past keyframes, the loop closure constraints, formulated as 4-DOF relative rigid body transforms, will also be added to the pose graph. The pose graph is optimized using all relative pose constraints (Sect. VIII-B) in a separate thread, and the relocalization module always runs with respect to the newest pose graph configuration.

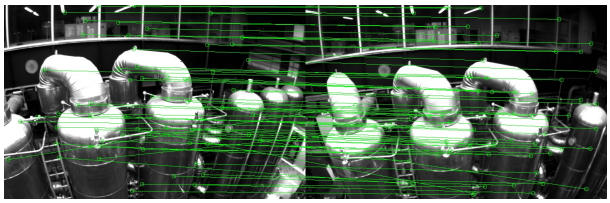
B. Feature Retrieval

When a loop is detected, the connection between the local sliding window and the loop closure candidate is established by retrieving feature correspondences. Correspondences are found by BRIEF descriptor matching. Directly descriptor matching may cause a large number a lot of outliers. To this end, we use two-step geometric outlier rejection, as illustrated in Fig. 10.

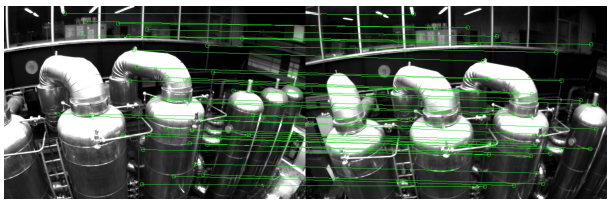
- 2D-2D: fundamental matrix test with RANSAC [31]. We use 2D observations of retrieved features in the current image and loop closure candidate image to perform fundamental matrix test.
- 3D-2D: PnP test with RANSAC [35]. Based on the known 3D position of features in the local sliding window, and 2D observations in the loop closure candidate image, we perform PnP test.



(a) BRIEF descriptor matching results



(b) First step: 2D-2D outlier rejection results



(c) Second step: 3D-2D outlier rejection results.

Fig. 10. Descriptor matching and outlier removal for feature retrieval during loop closure.

When the number of inliers beyond a certain threshold, we treat this candidate as a correct loop detection and perform relocalization.

C. Tightly-Coupled Relocalization

The relocalization process effectively aligns the current sliding window maintained by the monocular VIO (Sect. VI) to the graph of past poses. During relocalization, we treat poses of all loop closure frames as constants. We jointly optimize the sliding window using all IMU measurements, local visual measurement measurements, and retrieved feature correspondences from loop closure. We can easily write the visual measurement model for retrieved features observed by a loop closure frame v to be the same as those for visual measurements in VIO, as shown in (25). The only difference is that the pose $(\hat{\mathbf{q}}_v^w, \hat{\mathbf{p}}_v^w)$ of the loop closure frame, which is taken from the pose graph (Sect. VIII), or directly from past odometry output (if this is the first relocalization), is treated as a constant. To this end, we can slightly modify the nonlinear cost function in (22) with additional loop terms:

$$\min_{\mathcal{X}} \Bigg\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k\in\mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) \right\|^2_{\mathbf{P}_{b_{k+1}}^{b_k}} + \sum_{(l,j)\in\mathcal{C}} \rho\Big(\left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) \right\|^2_{\mathbf{P}_l^{c_j}} \Big) + \sum_{(l,v)\in\mathcal{L}} \rho(\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^v, \mathcal{X}, \hat{\mathbf{q}}_v^w, \hat{\mathbf{p}}_v^w)\|^2_{\mathbf{P}_l^{c_v}}) \Bigg\}, \tag{26}$$

where $\mathcal{L}$ is the set of the observation of retrieved features in the loop closure frames. (l, v) means l^{th} feature observed in

the loop closure frame v. Note that although the cost function is slightly different from (22), the dimension of the states to be solved remains the same, as poses of loop closure frames are considered as constants. When multiple loop closures are established with the current sliding window, we optimize using all loop closure feature correspondences from all frames at the same time. This gives multi-view constraints for relocalization, resulting in higher accuracy and better smoothness. Note that the global optimization of past poses and loop closure frames happens *after* relocalization, and will be discussed in Sect. VIII.

VIII. Global Pose Graph Optimization

After relocalization, the local sliding window shifts and aligns with past poses. Utilizing the relocalization results, this additional pose graph optimization step is developed to ensure the set of past poses are registered into a globally consistent configuration.

Since our visual-inertial setup renders roll and pitch angles fully observable, the accumulated drift only occurs in four degrees-of-freedom (x, y, z and yaw angle). To this end, we ignore estimating the drift-free roll and pitch states, and only perform 4-DOF pose graph optimization.

A. Adding Keyframes into the Pose Graph

When a keyframe is marginalized out from the sliding window, it will be added to pose graph. This keyframe serves as a vertex in the pose graph, and it connects with other vertexes by two types of edges:

1) Sequential Edge: a keyframe will establish several sequential edges to its previous keyframes. A sequential edge represents the relative transformation between two keyframes in the local sliding window, which value is taken directly from VIO. Considering a newly marginalized keyframe i and one of its previous keyframes j, the sequential edge only contains relative position $\hat{\mathbf{p}}_{ij}^{i}$ and yaw angle $\hat{\psi}_{ij}$.

$$\begin{aligned}\hat{\mathbf{p}}_{ij}^{i} =& \hat{\mathbf{R}}_{i}^{w\,-1}(\hat{\mathbf{p}}_{j}^{w} - \hat{\mathbf{p}}_{i}^{w}) \\ \hat{\psi}_{ij} =& \hat{\psi}_{j} - \hat{\psi}_{i}.\end{aligned} \tag{27}$$

2) Loop Closure Edge: If the newly marginalized keyframe has a loop connection, it will be connected with the loop closure frame by a loop closure edge in the pose graph. Similarly, the loop closure edge only contains 4-DOF relative pose transform that is defined the same as (27). The value of the loop closure edge is obtained using results from relocalization.

B. 4-DOF Pose Graph Optimization

We define the residual of the edge between frames i and j minimally as:

$$\mathbf{r}_{i,j}(\mathbf{p}_{i}^{w}, \psi_{i}, \mathbf{p}_{j}^{w}, \psi_{j}) = \begin{bmatrix} \mathbf{R}(\hat{\phi}_{i}, \hat{\theta}_{i}, \psi_{i})^{-1}(\mathbf{p}_{j}^{w} - \mathbf{p}_{i}^{w}) - \hat{\mathbf{p}}_{ij}^{i} \\ \psi_{j} - \psi_{i} - \hat{\psi}_{ij} \end{bmatrix}, \tag{28}$$

where $\hat{\phi}_{i}, \hat{\theta}_{i}$ are the estimates roll and pitch angles, which are obtained directly from monocular VIO.

The whole graph of sequential edges and loop closure edges are optimized by minimizing the following cost function:

$$\min_{\mathbf{p},\psi} \left\{ \sum_{(i,j)\in\mathcal{S}} \|\mathbf{r}_{i,j}\|^{2} + \sum_{(i,j)\in\mathcal{L}} \rho(\|\mathbf{r}_{i,j}\|^{2}) \right\}, \tag{29}$$

where $\mathcal{S}$ is the set of all sequential edges and $\mathcal{L}$ is the set of all loop closure edges. Although the tightly-coupled relocalization already helps with eliminating wrong loop closures, we add another Huber norm $\rho(\cdot)$ to further reduce the impact of any possible wrong loops. In contrast, we do not use any robust norms for sequential edges, as these edges are extracted from VIO, which already contain sufficient outlier rejection mechanisms.

Pose graph optimization and relocalization (Sect. VII-C) runs asynchronously in two separate threads. This enables immediate use of the most optimized pose graph for relocalization whenever it becomes available. Similarly, even if the current pose graph optimization is not completed yet, relocalization can still take place using the existing pose graph configuration. This process is illustrated in Fig. 9(b).

C. Pose Graph Management

The size of the pose graph may grow unbounded when the travel distance increases, limiting the real-time performance of the system in the long run. To this end, we implement a downsample process to maintain the pose graph database to a limited size. All keyframes with loop closure constraints will be kept, while other keyframes that are either too close or have very similar orientations to its neighbors may be removed. The probability of a keyframe being removed is proportional to its spatial density to its neighbors.

IX. Experimental Results

We perform three experiments and two applications to evaluate the proposed VINS-Mono system. In the first experiment, we compare the proposed algorithm with another state-of-the-art algorithm on public datasets. We perform a numerical analysis to show the accuracy of our system. We then test our system in the indoor environment to evaluate the performance in repetitive scenes. A large-scale experiment is carried out to illustrate the long-time practicability of our system. Additionally, we apply the proposed system for two applications. For aerial robot application, we use VINS-Mono for position feedback to control a drone to follow a pre-defined trajectory. We then port our approach onto an iOS mobile device and compare against Google Tango.

A. Dataset Comparison

We evaluate our proposed VINS-Mono using the EuRoC MAV Visual-Inertial Datasets [41]. The datasets are collected onboard a micro aerial vehicle, which contains stereo images (Aptina MT9V034 global shutter, WVGA monochrome, 20 FPS), synchronized IMU measurements (ADIS16448, 200 Hz), and ground truth states (VICON and Leica MS50). We only use images from the left camera. Large IMU bias and illumination changes are observed in these datasets.

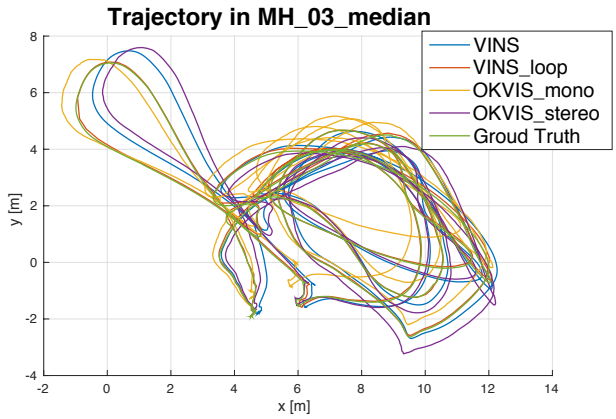


Fig. 11. Trajectory in MH_03_median, compared with OKVIS.

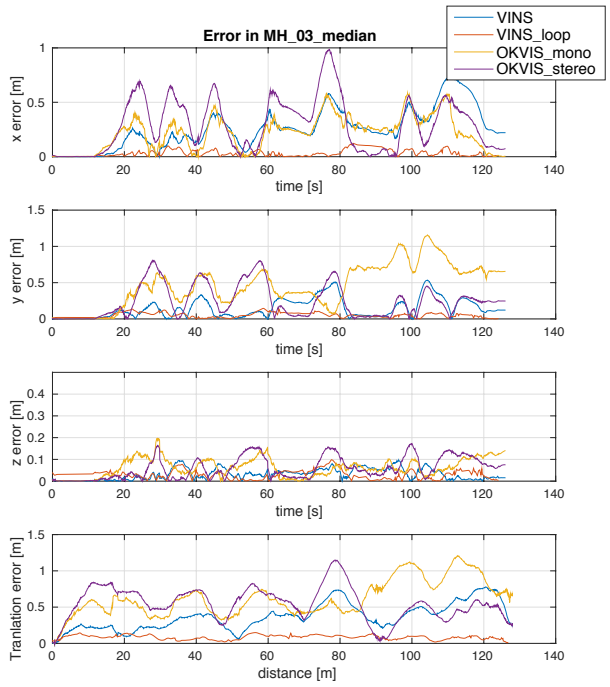


Fig. 12. Translation error plot in MH_03_median.

In these experiments, we compare VINS-Mono with OKVIS [16], a state-of-the-art VIO that works with monocular and stereo cameras. OKVIS is another optimization-based sliding-window algorithm. Our algorithm is different with OKVIS in many details, as presented in the technical sections. Our system is complete with robust initialization and loop closure. We use two sequences, MH_03_median and MH_05_difficult, to show the performance of proposed method. To simplify the notation, we use VINS to denote our approach with only monocular VIO, and VINS_loop to denote the complete version with relocalization and pose graph optimization. We use OKVIS_mono and OKVIS_stereo to denote the OKVIS's results using monocular and stereo images respectively. For the fair comparison, we throw the first 100 outputs, and use the following 150 outputs to align with the ground truth, and compare the remaining estimator outputs.

For the sequence MH_03_median, the trajectory is shown

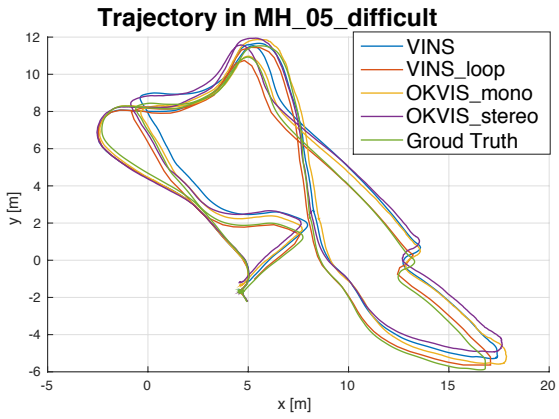


Fig. 13. Trajectory in MH_05_difficult, compared with OKVIS..

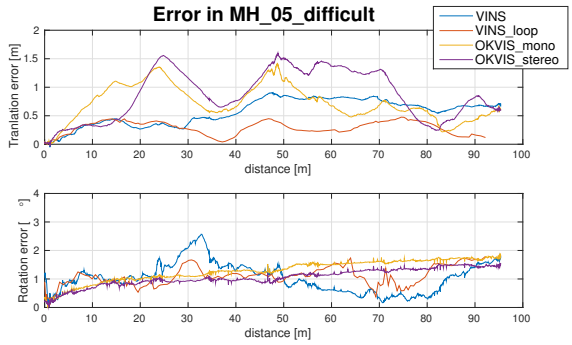


Fig. 14. Translation error and rotation error plot in MH_05_difficult.

in Fig. 11. We only compare translation error since rotated motion is negligible in this sequence. The x, y, z error versus time, and the translation error versus distance are shown in Fig. 12. In the error plot, VINS-Mono with loop closure has the smallest translation error. We observe similar results in MH_05_difficult. The proposed method with loop function has the smallest translation error. The translation and rotation errors are shown in Fig. 14. Since the movement is smooth without much yaw angle change in this sequence, only position drift occurs. Obviously, the loop closure capability efficiently bound the accumulated drifts. OKVIS performs better in roll and pitch angle estimation. A possible reason is that VINS-Mono uses the pre-integration technique which is the first-order approximation of IMU propagation to save computation resource.

VINS-Mono performs well in all EuRoC datasets, even in the most challenging sequence, V1_03_difficult, the one includes aggressive motion, texture-less area, and significant illumination change. The proposed method can initialize quickly in V1_03_difficult, due to the dedicated initialization procedure.

For pure VIO, both VINS-Mono and OKVIS have similar accuracy, it is hard to distinguish which one is better. However, VINS-Mono outperforms OKVIS at the system level. It is a complete system with robust initialization and loop closure function to assist the monocular VIO.

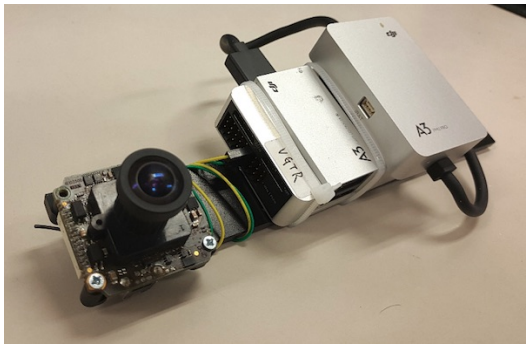
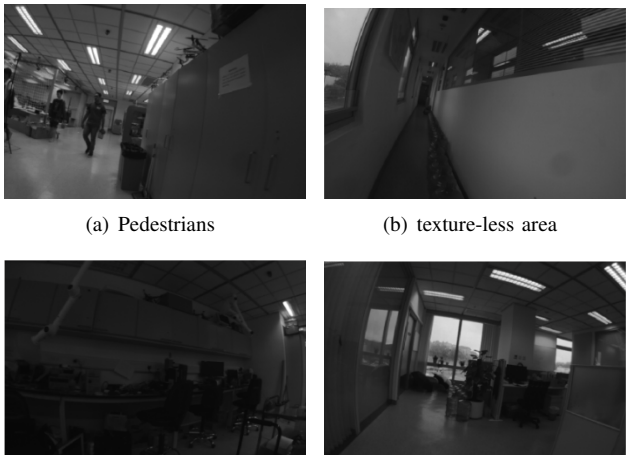


Fig. 15. The device we used for the indoor experiment. It contains one forward-looking global shutter camera (MatrixVision mvBlueFOX-MLC200w) with 752×480 resolution. We use the built-in IMU (ADXL278 and ADXRS290, 100Hz) for the DJI A3 flight controller.



(a) Pedestrians (b) texture-less area

(c) Low light condition (d) Glass and reflection

Fig. 16. Sample images for the challenging indoor experiment.

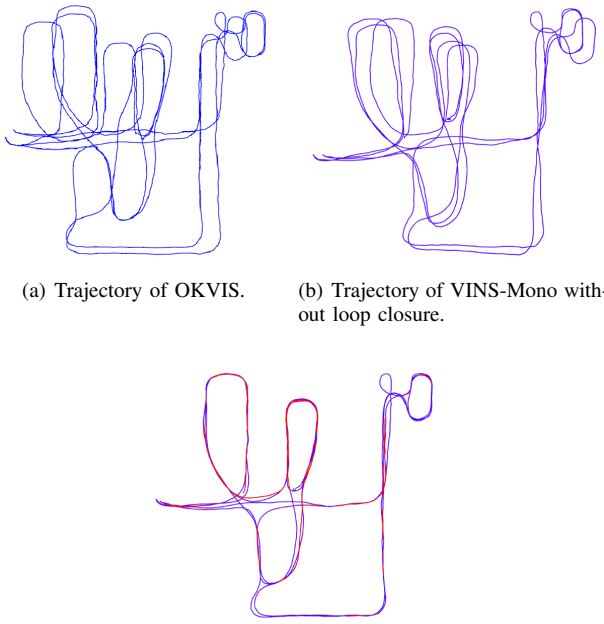
B. Indoor Experiment

In the indoor experiment, we choose our laboratory environment as the experiment area. The sensor suite we use is shown in Fig. 15. It contains a monocular camera (20Hz) and an IMU (100Hz) inside the DJI A3 controller[3]. We hold the sensor suite by hand and walk in normal pace in the laboratory. We encounter pedestrians, low light condition, texture-less area, glass and reflection, as shown in Fig. 16. Videos can be found in the multimedia attachment.

We compare our result with OKVIS, as shown in Fig. 17. Fig. 17(a) is the VIO output from OKVIS. Fig. 17(b) is the VIO-only result from proposed method without loop closure. Fig. 17(c) is the result of the proposed method with relocalization and loop closure. Noticeable VIO drifts occurred when we circle indoor. Both OKVIS and the VIO-only version of VINS-Mono accumulate significant drifts in x, y, z, and yaw angle. Our relocalization and loop closure modules efficiently eliminate all these drifts.

C. Large-scale Environment

1) Go out of the lab: We test VINS-Mono in a mixed indoor and outdoor setting. The sensor suite is the same as the one shown in Fig. 15. We started from a seat in the laboratory and go around the indoor space. Then we went down the stairs and walked around the playground outside the building. Next, we went back to the building and go upstairs. Finally, we returned to the same seat in the laboratory. The whole trajectory is more than 700 meters and last approximately ten minutes. A video of the experiment can be found in the multimedia attachment.



(a) Trajectory of OKVIS. (b) Trajectory of VINS-Mono without loop closure.

(c) Trajectory of VINS-Mono with relocalization and loop closure. Red lines indicate loop detection.

Fig. 17. Results of the indoor experiment with comparison against OKVIS.

The trajectory is shown in Fig. 19. Fig. 19(a) is the trajectory from OKVIS. When we went up stairs, OKVIS shows unstable feature tracking, resulting in bad estimation. We cannot see the shape of stairs in the red block. The VIO-only result from VINS-Mono is shown in Fig. 19(b). The trajectory with loop closure is shown in Fig. 19(c). The shape of stairs is clear in proposed method. The closed loop trajectory is aligned with Google Map to verify its accuracy, as shown in Fig. 18.

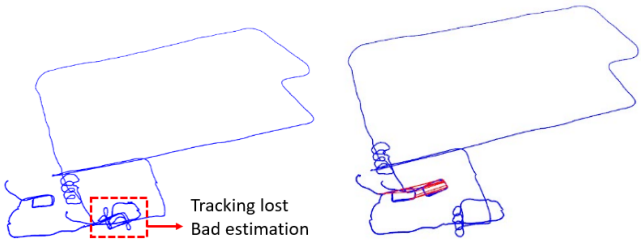
The final drift for OKVIS is [13.80, -5.26. 7.23]m in x, y and z-axis. The final dirft of VINS-Mono without loop closure is [-5.47, 2.76. -0.29]m, which occupies 0.88% with respect to the total trajectory length, smaller than OKVIS 2.36%. With loop correction. the final drift is bounded to [-0.032, 0.09, -0.07]m, which is trivial compared to the total trajectory length. Although we do not have ground truth, we can still visually inspect that the optimized trajectory is smooth and can be precisely aligned with the satellite map.

2) Go around campus: This very large-scale dataset that goes around the whole HKUST campus was recorded with a handheld VI-Sensor[4]. The dataset covers the ground that is around 710m in length, 240m in width, and with 60m in height changes. The total path length is 5.62km. The data contains the 25Hz image and 200Hz IMU lasting for 1 hour and 34

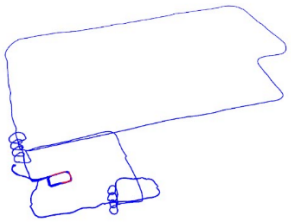
[3]http://www.dji.com/a3

[4]http://www.skybotix.com/

Fig. 18. The estimated trajectory of the mixed indoor and outdoor experiment aligned with Google Map. The yellow line is the final estimated trajectory from VINS-Mono after loop closure. Red lines indicate loop closure.



(a) Estimated trajectory from OKVIS (b) Estimated trajectory from VINS-Mono with loop closure disabled



(c) Estimated trajectory from VINS-Mono with loop closure

Fig. 19. Estimated trajectories for the mixed indoor and outdoor experiment. In Fig. 19(a), results from OKVIS went bad during tracking lost in texture-less region (staircase). Figs. 19(b) 19(c) shows results from VINS-Mono without and with loop closure. Red lines indicate loop closures. The spiral-shaped blue line shows the trajectory when going up and down the stairs. We can see that VINS-Mono performs well (subject to acceptable drift) even without loop closure.

minutes. It is a very significant experiment to test the stability and durability of VINS-Mono.

In this large-scale test, We set the keyframe database size to 2000 in order to provide sufficient loop information and achieve real-time performance. We run this dataset with an Intel i7-4790 CPU running at 3.60GHz. Timing statistics are show in Table. I. The estimated trajectory is aligned with Google map in Fig. 20. Compared with Google map, we can see our results are almost drift-free in this very long-duration test.

D. Application I: Feedback Control of an Aerial Robot

We apply VINS-Mono for autonomous feedback control of an aerial robot, as shown in Fig. 21(a). We use a forward-looking global shutter camera (MatrixVision mvBlueFOX-MLC200w) with 752×480 resolution, and equipped it with a 190-degree fisheye lens. A DJI A3 flight controller is used for both IMU measurements and for attitude stabilization control. The onboard computation resource is an Intel i7-5500U CPU running at 3.00 GHz. Traditional pinhole camera model is not suitable for large FOV camera. We use MEI [42] model for this camera, calibrated by the toolkit introduced in [43].

In this experiment, we test the performance of autonomous trajectory tracking under using state estimates from VINS-Mono. Loop closure is disabled for this experiment The quadrotor is commanded to track a figure eight pattern with each circle being 1.0 meters in radius, as shown in Fig. 21(b). Four obstacles are put around the trajectory to verify the accuracy of VINS-Mono without loop closure. The quadrotor follows this trajectory four times continuously during the experiment. The 100 Hz onboard state estimates (Sect. VI-F) enables real-time feedback control of quadrotor.

Ground truth is obtained using OptiTrack[5]. Total trajectory length is 61.97 m. The final drift is [0.08, 0.09, 0.13] m, resulting in 0.29% position drift. Details of the translation and rotation as well as their corresponding errors are shown in Fig. 23.

E. Application II: Mobile Device

We port VINS-Mono to mobile devices and present a simple AR application to showcase its accuracy and robustness. We name our mobile implementation VINS-Mobile[6], and we compare it against Google Tango device[7], which is one of the best commercially available augmented reality solutions on mobile platforms.

VINS-Mono runs on an iPhone7 Plus. we use 30 Hz images with 640 × 480 resolution captured by the iPhone, and IMU data at 100 Hz obtained by the built-in InvenSense MP67B 6-axis gyroscope and accelerometer. As Fig. 24 shows, we mount the iPhone together with a Tango-Enabled Lenovo Phab 2 Pro. The Tango device uses a global shutter fisheye camera and synchronized IMU for state estimation. Firstly, we insert a virtual cube on the plane which is extracted from estimated visual features as shown in Fig. 25(a). Then we hold these

[5]http://optitrack.com/
[6]https://github.com/HKUST-Aerial-Robotics/VINS-Mobile
[7]http://shopap.lenovo.com/hk/en/tango/

TABLE I
TIMING STATISTICS

Tread	Modules	Time (ms)	Rate (Hz)
1	Feature detector KLT tracker	15 5	25 25
2	Window optimization	50	10
3	Loop detection Pose graph optimization	100 130	

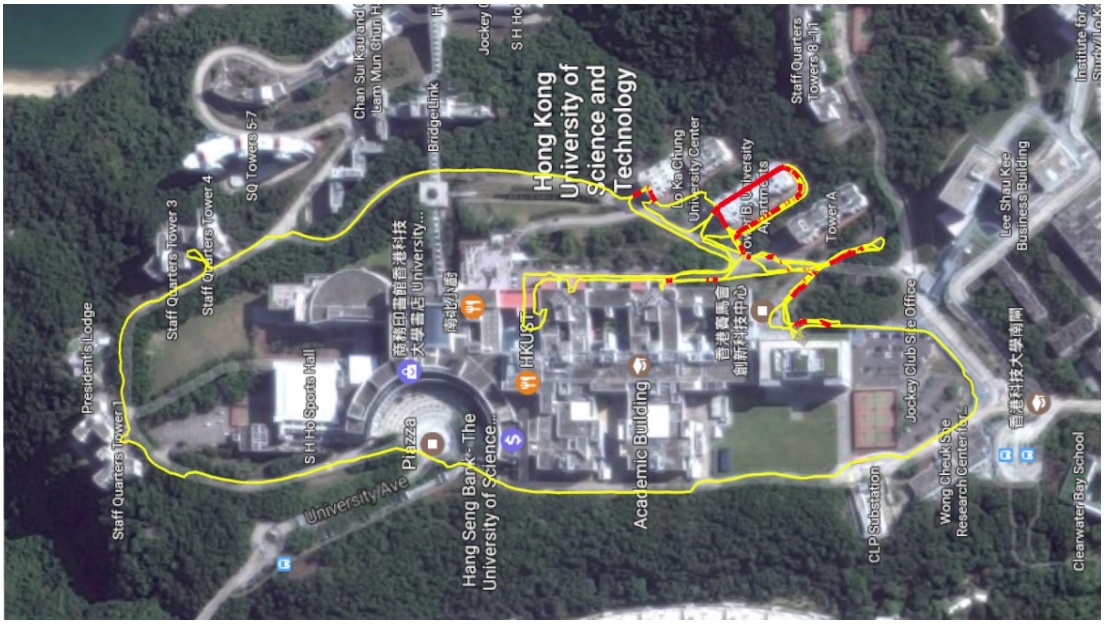
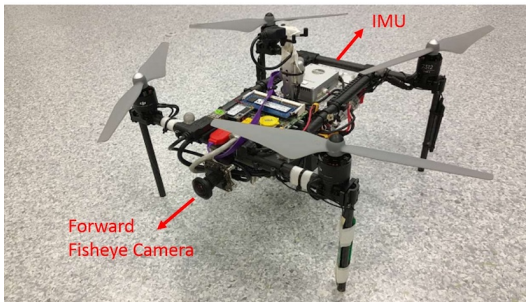
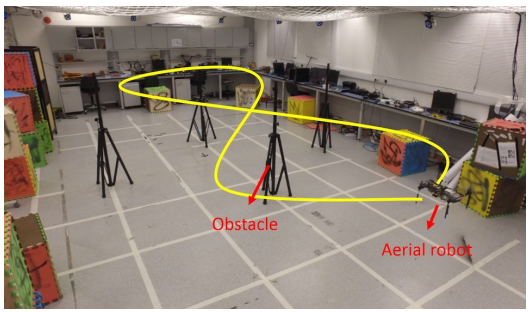


Fig. 20. The estimated trajectory of the very large-scale environment aligned with Google map. The yellow line is the estimated trajectory from VINS-Mono. Red lines indicates loop closure.



(a) Aerial robot testbed



(b) Testing environment and desired figure eight pattern

Fig. 21. Fig 21(a): The self-developed aerial robot with a forward-looking fisheye camera (MatrixVision mvBlueFOX-MLC200w, 190 FOV) and an DJI A3 flight controller (ADXL278 and ADXRS290, 100Hz). Fig. 21(b): The designed trajectory. Four known obstacles are placed. The yellow line is the predefined figure eight-figure pattern which the aerial robot should follow. The robot follows the trajectory four times with loop closure disabled. A video of the experiment can be found in the multimedia attachment.

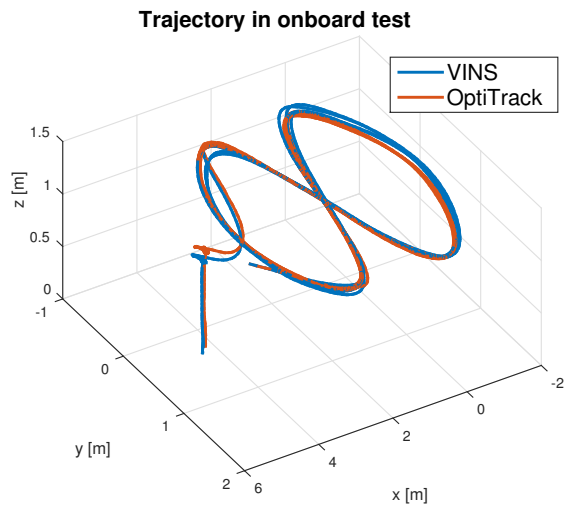


Fig. 22. The trajectory of loop closure-disabled VINS-Mono on the MAV platform and its comparison against the ground truth. The robot follows the trajectory four times. VINS-Mono estimates are used as real-time position feedback for the controller. Ground truth is obtained using OptiTrack. Total length is $61.97m$. Final drift is $0.18m$.

two devices and walk inside and outside the room in a normal pace. When loops are detected, we use the 4-DOF pose graph optimization (Sect. VIII-B) to eliminate x, y, z and yaw drifts.

Interestingly, when we open a door, Tango's yaw estimation jumps a big angle, as shown in Fig. 25(b). The reason maybe estimator crash caused by unstable feature tracking or active failure detection and recovery. However, VINS-Mono still works well in this challenging case. After traveling about 264m, we return to the start location. The final result can be seen in Fig. 25(c), the trajectory of Tango suffers drifting in the last lap while our VINS returns to the start point. The drift in total trajectory is eliminated due to the 4-DOF pose graph optimization. This is also evidenced by the fact that the cube is registered to the same place on the image comparing to the beginning.

Admittedly, Tango is more accurate than our implementation especially for local state estimates. However, this exper-

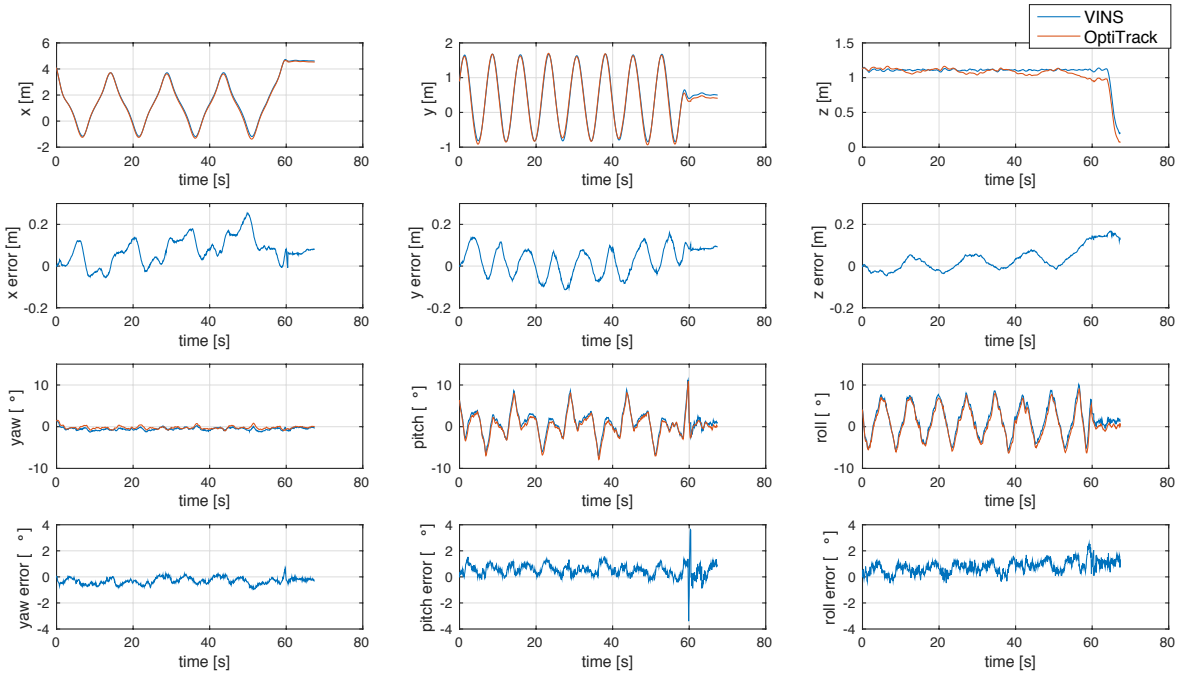


Fig. 23. Position, orientation and their corresponding errors of loop closure-disabled VINS-Mono compared with OptiTrack. A sudden in pitch error at the 60s is caused by aggressive breaking at the end of the designed trajectory, and possible time misalignment error between VINS-Mono and OptiTrack.

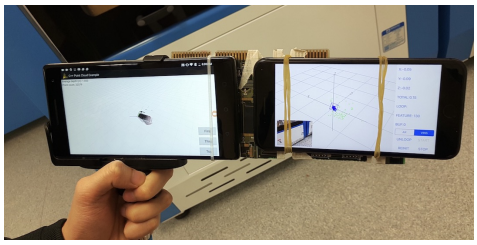


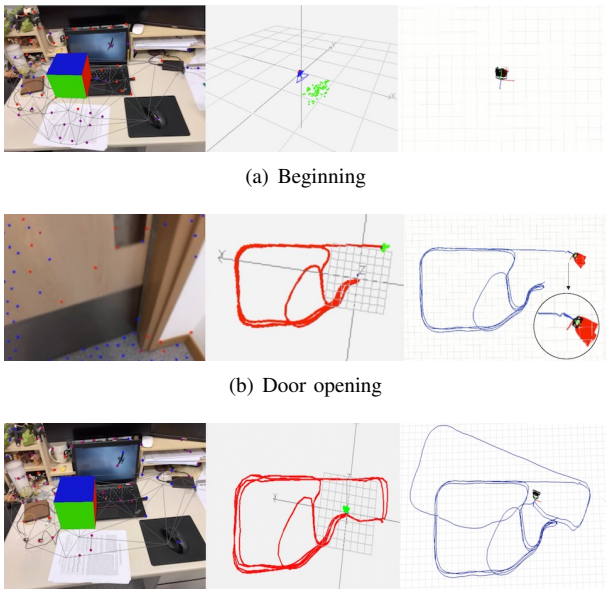
Fig. 24. A simple holder that we used to mount the Google Tango device (left) and the iPhone7 Plus (right) that runs our VINS-Mobile implementation.

iment shows that the proposed method can run on general mobile devices and have the potential ability to compare specially engineered devices. The robustness of proposed method is also proved in this experiment. Video can be found in the multimedia attachment.

X. Conclusion and Future Work

In this paper, we propose a robust and versatile monocular visual-inertial estimator. Our approach features both state-of-the-art and novel solutions to IMU pre-integration, estimator initialization and failure recovery, online extrinsic calibration, tightly-coupled visual-inertial odometry, relocalization, and efficient global optimization. We show superior performance by comparing against both state-of-the-art open source implementations and highly optimized industry solutions. We open source both PC and iOS implementation for the benefit of the community.

Although feature-based VINS estimators have already reached the maturity of real-world deployment, we still see many directions for future research. Monocular VINS may



(a) Beginning

(b) Door opening

(c) End

Fig. 25. From left to right: AR image from VINS-Mobile, estimated trajectory from VINS-Mono, estimated trajectory from Tango Fig: 25(a): Both VINS-Mobile and Tango are initialized at the start location and a virtual box is inserted on the plane which extracted from estimated features. Fig. 25(b): A challenging case in which the camera is facing a moving door. The drift of Tango trajectory is highlighted. Fig. 25(c): The final trajectory of both VINS-Mobile and Tango. The total length is about 264m.

reach weakly observable or even degenerate conditions depending on the motion and the environment. We are most interested in online methods to evaluate the observability properties of monocular VINS, and online generation of motion plans to restore observability. Another research direction concerns the

mass deployment of monocular VINS on a large variety of consumer devices, such as mobile phones. This application requires online calibration of almost all sensor intrinsic and extrinsic parameters, as well as the online identification of calibration qualities. Finally, we are interested in producing dense maps given results from monocular VINS. Our first results on monocular visual-inertial dense mapping with application to drone navigation was presented in [44]. However, extensive research is still necessary to further improve the system accuracy and robustness.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Hong Kong, China, May 2014.

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer International Publishing, 2014, pp. 834–849.

[4] R. Mur-Artal, J. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[6] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[7] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

[8] Z. Yang and S. Shen, "Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 39–51, 2017.

[9] T. Qin and S. Shen, "Robust initialization of monocular visual-inertial estimation on aerial robots." in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, 2017, accepted.

[10] P. Li, T. Qin, B. Hu, F. Zhu, and S. Shen, "Monocular visual-inertial state estimation for mobile augmented reality." in *Proc. of the IEEE Int. Sym. on Mixed adn Augmented Reality*, Nantes, France, 2017, accepted.

[11] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 957–964.

[12] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.* IEEE, 2013, pp. 3923–3929.

[13] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Roma, Italy, Apr. 2007, pp. 3565–3572.

[14] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robot. Research*, vol. 32, no. 6, pp. 690–711, May 2013.

[15] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Syst.* IEEE, 2015, pp. 298–304.

[16] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Int. J. Robot. Research*, vol. 34, no. 3, pp. 314–334, Mar. 2014.

[17] R. Mur-Artal and J. D. Tardos, "Visual-inertial monocular SLAM with map reuse," *arXiv preprint arXiv:1610.05949*, 2016.

[18] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices." in *Robotics: Science and Systems*, 2015.

[19] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo vins," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.*, Singapore, May 2017.

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[21] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2016, pp. 1885–1892.

[22] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.

[23] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, Jul. 2015.

[24] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *Proc. of the Int. Sym. on Exp. Robot.*, Marrakech, Morocco, Jun. 2014.

[25] A. Martinelli, "Closed-form solution of visual-inertial structure from motion," *Int. J. Comput. Vis.*, vol. 106, no. 2, pp. 138–152, 2014.

[26] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 18–25, 2017.

[27] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2015, pp. 1722–1729.

[28] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," *Robotics: Science and Systems VI*, 2010.

[29] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, Vancouver, Canada, Aug. 1981, pp. 24–28.

[30] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[31] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[32] A. Heyden and M. Pollefeys, "Multiple view geometry," *Emerging Topics in Computer Vision*, 2005.

[33] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[34] T. Liu and S. Shen, "Spline-based initialization of monocular visual-inertial state estimators at high altitude," *IEEE Robotics and Automation Letters*, 2017, accepted.

[35] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[36] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmenta modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[37] P. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 73–101, 1964.

[38] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[39] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *J. Field Robot.*, vol. 27, no. 5, pp. 587–608, Sep. 2010.

[40] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[41] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

[42] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3945–3950.

[43] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1793–1800.

[44] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 00, pp. 1–29, 2017.