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

Tong Qin, Peiliang Li, Zhenfei Yang, Shaojie Shen

***Abstract*—This paper presents a monocular visual-inertial system (VINS) for state estimation in various environments. The monocular camera and a low-cost inertial measurement unit (IMU) constitute the minimum sensor suite for six degrees-of-freedom state estimation. Our algorithm optimizes the visual and inertial measurements in a bounded sliding window iteratively to perform accurate state estimation. Visual structure is maintained by keyframes in the sliding window, while inertial metric measurements are kept by pre-integration between keyframes. Our system is robust to initialization with unknown states, online camera-IMU extrinsic parameter calibration, unified reprojection error defined on the sphere, loop detection, and four degrees-of-freedom pose graph optimization. These properties make our system practical and easy to use. We validate the performance of our system on the public dataset and real-world experiments by comparing 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 application. We highlight that the proposed work is a reliable and complete system, which can be easily operated on other intelligent platforms. We open source our implementation code[1] and an augmented reality (AR) application on iOS mobile devices[2].**

I. Overview

The structure of proposed visual-inertial system is shown in Fig. **??**. The first part is measurements processing front-end which extracts, tracks features for each new image frame and pre-integrates all the IMU data between two frames. The second part is initialization procedure, which provides necessary initial values (pose, velocity, gravity vector, gyroscope bias and 3D feature position) to bootstrap the nonlinear system. The third part implements nonlinear graph optimization to solve the states in our sliding window by optimizing all the visual, inertial information together. Another part which runs in another thread takes charge of loop detection and pose graph optimization.

Notation: We consider $(\cdot)^w$ as world frame, where gravity vector is along with z axis. $(\cdot)^b$ is body frame which is aligned with IMU frame. $(\cdot)^c$ is camera frame. We use both $\mathbf{R}$ and quaternion $\mathbf{q}$ to denote rotation matrix. The quaternion corresponds to the Hamilton notation. We use quaternion in the states, while we use its corresponding rotation matrix in some equations where it multiplies a vector. $\mathbf{q}_b^w, \mathbf{p}_b^w$ are the rotation and translation from body frame to 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. $\otimes$ is the two quaternion multiplication operation. $\mathbf{g}^w = [0, 0, g]^T$ is the gravity vector in the world frame.

T. Qin, P. Li and S. Shen are with Robotics Institute, Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology. Z. Yang is with Dajiang Innovations Technology Co., Ltd. e-mail: {tong.qin, pliap, zyangag}@connect.ust.hk, eeshaojie@ust.hk

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

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

II. Measurement Preprocessing

We pre-process visual and inertial measurements in this section. For visual measurements, we track features between consecutive frames and detect new features in latest frame. For IMU measurements, we pre-integrate them between two consecutive frames. Note that IMU measurement is affected by both bias and noise. So we especially take bias into consideration in IMU pre-integration and optimization part, which is essential for low-cost IMU chips.

A. Vision Processing Front-end

For each new image, the existing features are tracked by the KLT sparse optical flow algorithm [26]. Meanwhile, new corner features are detected [27] 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. Features are projected to a unit sphere after passing outlier rejection. Outlier rejection is performed by the RANSAC step in the fundamental matrix test.

Keyframes are also selected in this step. We have two criteria for keyframe selection. One of them is average parallax. If the average parallax of the tracked features is beyond a certain threshold, we treat this image as a 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 the IMU propagation result to compensate the rotation when calculating the parallax. Another criterion is tracking quality. If the number of tracked features goes below a certain threshold, we treat this frame also as a keyframe.

B. IMU Pre-integration

We extend the IMU pre-integration proposed in our previous work [7] by incorporating IMU bias correction [28]. Compared with [28], we derive noise propagation in continuous-time dynamics. In addition, the IMU pre-integration results are also used in initialization procedure to calibrate initial states.

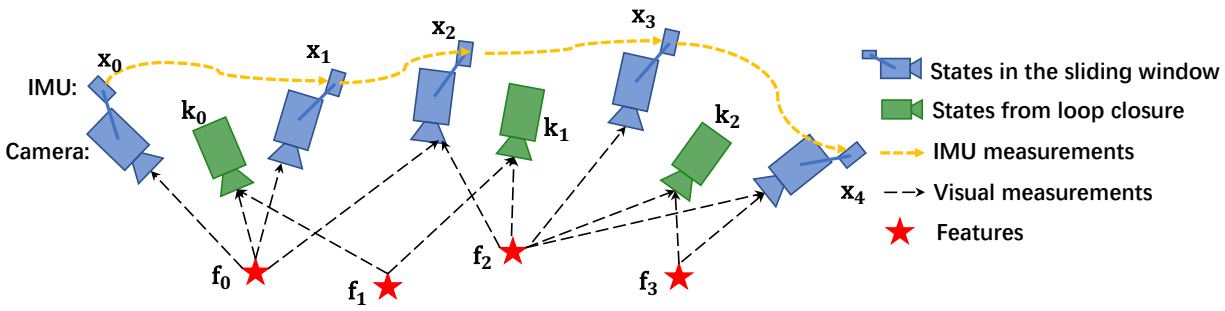


Fig. 1: An illustration of our sliding window tightly coupled with IMU, visual and loop measurements.

Given two time instants that correspond to images frame b_k and b_{k+1}, the state variables are constrained by inertial measurements during time interval $[k, k+1]$:

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

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2}$$

Δt_k is the duration of time interval $[k, k+1]$. $\hat{\boldsymbol{\omega}}_t$, $\hat{\mathbf{a}}_t$ are raw IMU measurements, which are in body frame and affected by acceleration bias $\mathbf{b}_a$, gyroscope bias $\mathbf{b}_w$, and noise.

It can be seen that the IMU state propagation require the 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 the IMU measurements between them. This propagation strategy is computation consuming. To avoid re-propagation, we adopt pre-integration algorithm.

After change the reference frame for IMU propagation to 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:

$$\begin{aligned}
\mathbf{R}^{b_k}_w \mathbf{p}^w_{b_{k+1}} &= \mathbf{R}^{b_k}_w(\mathbf{p}^w_{b_k} + \mathbf{v}^w_{b_k}\Delta t_k - \frac{1}{2}\mathbf{g}^w\Delta t_k^2) + \boldsymbol{\alpha}^{b_k}_{b_{k+1}} \\
\mathbf{R}^{b_k}_w \mathbf{v}^w_{b_{k+1}} &= \mathbf{R}^{b_k}_w(\mathbf{v}^w_{b_k} - \mathbf{g}^w\Delta t_k) + \boldsymbol{\beta}^{b_k}_{b_{k+1}} \\
\mathbf{q}^{b_k}_w \otimes \mathbf{q}^w_{b_{k+1}} &= \boldsymbol{\gamma}^{b_k}_{b_{k+1}}
\end{aligned} \tag{3}$$

where,

$$\begin{aligned}
\boldsymbol{\alpha}^{b_k}_{b_{k+1}} &= \iint_{t\in[k,k+1]} \mathbf{R}^{b_k}_t(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt^2 \\
\boldsymbol{\beta}^{b_k}_{b_{k+1}} &= \int_{t\in[k,k+1]} \mathbf{R}^{b_k}_t(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt \\
\boldsymbol{\gamma}^{b_k}_{b_{k+1}} &= \int_{t\in[k,k+1]} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t})\boldsymbol{\gamma}^{b_k}_t dt,
\end{aligned} \tag{4}$$

$\boldsymbol{\gamma}^{b_k}_{b_k}$ is identity quaternion at the beginning. It can be seen that the pre-integration part (4) can be obtained solely with IMU measurements by taking b_k as the base frame. $\boldsymbol{\alpha}^{b_k}_{b_{k+1}}, \boldsymbol{\beta}^{b_k}_{b_{k+1}}, \boldsymbol{\gamma}^{b_k}_{b_{k+1}}$ are only related to IMU bias instead of other states in b_k and b_{k+1}. At the very beginning, acceleration bias and gyroscope bias are zero. When the estimation of bias changes, if the change is small, we adjust $\boldsymbol{\alpha}^{b_k}_{b_{k+1}}, \boldsymbol{\beta}^{b_k}_{b_{k+1}}, \boldsymbol{\gamma}^{b_k}_{b_{k+1}}$ by its first-order approximation with respect to bias, otherwise we do re-propagation. This strategy saves a lot of computation resource for optimization-based algorithm since we don't need to propagate IMU measurements again and again.

In the discrete-time implementation, various 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 our 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, which represents the identity matrix. The mean of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ is propagate step by step as follows,

$$\begin{aligned}
\hat{\boldsymbol{\alpha}}^{b_k}_{i+1} &= \hat{\boldsymbol{\alpha}}^{b_k}_i + \hat{\boldsymbol{\beta}}^{b_k}_i \delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{\gamma}}^{b_k}_i)(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t^2 \\
\hat{\boldsymbol{\beta}}^{b_k}_{i+1} &= \hat{\boldsymbol{\beta}}^{b_k}_i + \mathbf{R}(\hat{\boldsymbol{\gamma}}^{b_k}_i)(\hat{\mathbf{a}}_i - \mathbf{b}_{a_i})\delta t \\
\hat{\boldsymbol{\gamma}}^{b_k}_{i+1} &= \hat{\boldsymbol{\gamma}}^{b_k}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{w_i})\delta t \end{bmatrix}
\end{aligned}. \tag{5}$$

i and $i+1$ are two discrete moments corresponding to two IMU measurements within $[k, k+1]$. δt is the time interval between two IMU measurements i and $i+1$.

Then we deal with the covariance propagation. Assuming that the noise in acceleration and gyroscope measurements are Gaussian white noise, $\mathbf{n}_a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2_a)$, $\mathbf{n}_w \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2_w)$. The acceleration bias and gyroscope bias are random walk, whose derivatives are Gaussian white noise, $\mathbf{n}_{b_a} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2_{b_a})$, $\mathbf{n}_{b_w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2_{b_w})$:

$$\dot{\mathbf{b}}_{a_t} = \mathbf{n}_{b_a}, \quad \dot{\mathbf{b}}_{w_t} = \mathbf{n}_{b_w}. \tag{6}$$

Since $\boldsymbol{\gamma}_t^{b_k}$ is over-parameterized, we define its error terms as the perturbation

$$\boldsymbol{\gamma}_t^{b_k} \approx \hat{\boldsymbol{\gamma}}_t^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}_t^{b_k} \end{bmatrix}. \tag{7}$$

We can derive the continuous-time linearized dynamics of the error terms from eq 4 and eq 6:

$$\begin{bmatrix} \delta\dot{\boldsymbol{\alpha}}_t^{b_k} \\ \delta\dot{\boldsymbol{\beta}}_t^{b_k} \\ \delta\dot{\boldsymbol{\theta}}_t^{b_k} \\ \delta\dot{\mathbf{b}}_{a_t} \\ \delta\dot{\mathbf{b}}_{w_t} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{R}_t^{b_k}\lfloor \hat{\mathbf{a}}_t - \mathbf{b}_{a_t} \rfloor_\times & -\mathbf{R}_t^{b_k} & 0 \\ 0 & 0 & -\lfloor \hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t} \rfloor_\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_t^{b_k} \\ \delta\boldsymbol{\beta}_t^{b_k} \\ \delta\boldsymbol{\theta}_t^{b_k} \\ \delta\mathbf{b}_{a_t} \\ \delta\mathbf{b}_{w_t} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{R}_t^{b_k} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_w \\ \mathbf{n}_{b_a} \\ \mathbf{n}_{b_w} \end{bmatrix} = \mathbf{F}_t \delta\mathbf{z}_t^{b_k} + \mathbf{G}_t \mathbf{n}_t. \tag{8}$$

where $\lfloor \cdot \rfloor_\times$ is the skew-matrix operate. Details about error-state representation of quaternion can be found in [29]. $\mathbf{P}_{b_{k+1}}^{b_k}$ can be computed recursively by the first-order discrete-time covariance update with the initial covariance $\mathbf{P}_{b_k}^{b_k} = \mathbf{0}$:

$$\mathbf{P}_{t+\delta t}^{b_k} = (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{P}_t^{b_k}(\mathbf{I} + \mathbf{F}_t\delta t)^T + (\mathbf{G}_t\delta t)\mathbf{Q}(\mathbf{G}_t\delta t)^T, \quad t \in [k, k+1], \tag{9}$$

where δt is the time between two IMU measurements, and $\mathbf{Q}$ 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 $\mathbf{J}_{b_k} = \mathbf{I}$,

$$\mathbf{J}_{t+\delta t} = (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{J}_t, \quad t \in [k, k+1] \tag{10}$$

In this recursive way, 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 the bias can be write as:

$$\begin{aligned} \boldsymbol{\alpha}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\alpha}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\alpha}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\beta}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\beta}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\delta\mathbf{b}_{w_k} \end{bmatrix} \end{aligned} \tag{11}$$

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 $\mathbf{J}_{b_w}^{\alpha}, \mathbf{J}_{b_a}^{\beta}, \mathbf{J}_{b_w}^{\beta}, \mathbf{J}_{b_w}^{\gamma}$. Note that $\boldsymbol{\gamma}_{b_{k+1}}^{b_k}$ is only influenced by $\mathbf{b}_{w_k}$. When the estimation of bias change slightly, we use eq. 11 to correct pre-integration results approximately instead of re-propagation.

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

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k) \\ \mathbf{R}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w) \\ \mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^w \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k} \end{bmatrix}. \tag{12}$$

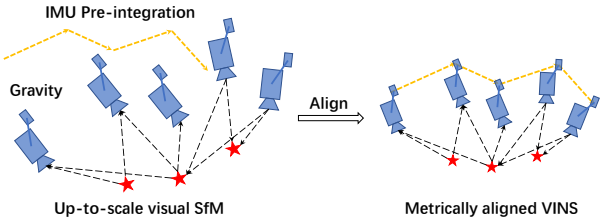


Fig. 2: An illustration of aligning IMU pre-integration with visual structure.

III. Initialization

Monocular tightly-coupled visual-inertial optimization 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 a good initial guess. In usual, under the stationary assumption, the average of IMU measurements in first few seconds is treated as gravity vector and IMU propagation is treated as the initial poses. However, this treatment is improper when IMU measurements are influenced by non-trivial bias or accelerated movement occurs at the beginning. To improve the success rate of the monocular visual-inertial system, a robust initialization procedure is required.

We adopt a loosely-coupled sensor fusion method to get the 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 a derived initial guess from relative motion method, such as the Eight-point [30], Five-point [31], homogeneous and fundamental method. Through align the metric IMU pre-integration into the visual-only structure, we can roughly recover the scale, gravity, velocity, and even bias, which is of benefit for bootstrapping a nonlinear system, as shown in Fig. 2.

A. Visual SfM in Sliding Window

The initialization procedure starts with a vision-only structure, which estimates a graph of up-to-scale camera poses and feature positions.

Firstly, we check the feature correspondences between the latest frame and previous frames. If we can find a previous frame which has more than 30 tracked features and the average parallax is more than 20 pixel between the latest frame, we recover the relative rotation and up-to-scale translation between these two frames by Five-point method [31]. Otherwise, we keep the latest frame in the window and wait for new frames. If

Five-point method success, we fix the scale of this translation and triangulate all the features observed in these two frames. Based on these triangulated features, Perspective-n-Point (PnP) method is performed to estimate poses of other frames in the whole window. Finally, a global full Bundle Adjustment [32] is applied to minimize the total re-projection error of all feature observations. Since we don't know the absolute world frame, we set the first camera frame $(\cdot)^{c_0}$ as the base frame. All frame poses $(\bar{\mathbf{p}}^{c_0}_{c_k}, \mathbf{q}^{c_0}_{c_k})$ and feature positions are represented with respect to $(\cdot)^{c_0}$. According to the prior of the extrinsic parameter $(\mathbf{p}^b_c, \mathbf{q}^b_c)$ between camera frame and IMU (body) frame, we can translate the poses from camera center to body center,

$$\begin{aligned} \mathbf{q}^{c_0}_{b_k} &= \mathbf{q}^{c_0}_{c_k} \otimes (\mathbf{q}^b_c)^{-1} \\ s\bar{\mathbf{p}}^{c_0}_{b_k} &= s\bar{\mathbf{p}}^{c_0}_{c_k} - \mathbf{R}^{c_0}_{b_k}\mathbf{p}^b_c, \end{aligned} \tag{13}$$

where s aligns the visual structure to the absolute scale, which will be solved in the following section.

B. Visual-Inertial Alignment

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

$$\begin{aligned} &\min_{\delta b_w} \sum_{k\in\mathcal{B}} \left\| {\mathbf{q}^{c_0}_{b_{k+1}}}^{-1} \otimes \mathbf{q}^{c_0}_{b_k} \otimes \boldsymbol{\gamma}^{b_k}_{b_{k+1}} \right\|^2 \\ &\boldsymbol{\gamma}^{b_k}_{b_{k+1}} \approx \hat{\boldsymbol{\gamma}}^{b_k}_{b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}^{\gamma}_{b_w}\delta\mathbf{b}_w \end{bmatrix}, \end{aligned} \tag{14}$$

where $\mathcal{B}$ indexes all frames in the window. The second equation is first order approximation of $\hat{\boldsymbol{\gamma}}^{b_k}_{b_{k+1}}$ with respect to the gyroscope bias in its error-state presentation. In such way, we get an initial calibration of the gyroscope bias $\mathbf{b}_w$. Then we re-propagate $\hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}}, \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}}$ using the new gyroscope bias.

2) Velocity, Gravity Vector and Metric Scale Initialization: We define the variables (velocity, gravity vector and metric scale) that we estimate in this step as

$$\mathcal{X}_I = \left[\mathbf{v}^{b_0}_{b_0}, \mathbf{v}^{b_1}_{b_1}, \cdots \mathbf{v}^{b_n}_{b_n}, \mathbf{g}^{c_0}, s\right], \tag{15}$$

where $\mathbf{v}^{b_n}_{b_n}$ is velocity in body frame at time b_n, $\mathbf{g}^{c_0}$ is the gravity vector under c_0 frame, and scale s is a scalar. Considering two consecutive frames b_k and b_{k+1} in the window, the IMU measurement model for position and velocity, expressed in the local frame b_k, can be written as:

$$\begin{aligned} \hat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}} &= \mathbf{R}^{b_k}_{c_0}(s(\bar{\mathbf{p}}^{c_0}_{b_{k+1}} - \bar{\mathbf{p}}^{c_0}_{b_k}) + \frac{1}{2}\mathbf{g}^{c_0}\Delta t_k^2 - \mathbf{R}^{c_0}_{b_k}\mathbf{v}^{b_k}_{b_k}\Delta t_k) \\ \hat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}} &= \mathbf{R}^{b_k}_{c_0}(\mathbf{R}^{c_0}_{b_{k+1}}\mathbf{v}^{b_{k+1}}_{b_{k+1}} + \mathbf{g}^{c_0}\Delta t_k - \mathbf{R}^{c_0}_{b_k}\mathbf{v}^{b_k}_{b_k}). \end{aligned} \tag{16}$$

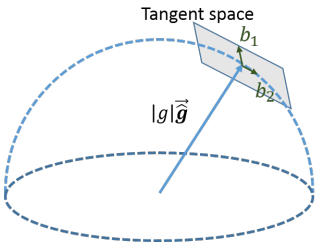


Fig. 3: Illustration of two degrees-of-freedom parameterization of gravity. Since the magnitude of gravity is known, $\mathbf{g}$ lies on a sphere with the radius g. The gravity is parameterized around current estimate as $g\cdot\bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, where $\mathbf{b}_1$ and $\mathbf{b}_2$ are two orthogonal basis spanning the tangent space.

We can rewrite eq.13 and eq.16 into the following linear form:

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

$$\begin{bmatrix} -\mathbf{I}\Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}^{b_k}_{c_0}\Delta t_k^2 & \mathbf{R}^{b_k}_{c_0}(\bar{\mathbf{p}}^{c_0}_{c_{k+1}} - \bar{\mathbf{p}}^{c_0}_{c_k}) \\ -\mathbf{I} & \mathbf{R}^{b_k}_{c_0}\mathbf{R}^{c_0}_{b_{k+1}} & \mathbf{R}^{b_k}_{c_0}\Delta t_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{b_k}_{b_k} \\ \mathbf{v}^{b_{k+1}}_{b_{k+1}} \\ \mathbf{g}^{c_0} \\ s \end{bmatrix} \tag{17}$$

In the above formula, $\mathbf{R}^{c_0}_{b_k}, \mathbf{R}^{c_0}_{b_{k+1}}, \bar{\mathbf{p}}^{c_0}_{c_k}, \bar{\mathbf{p}}^{c_0}_{c_{k+1}}$ are obtained from the visual structure. Δt_k is the time interval between two consecutive frames. By solving the this least square problem:

$$\min_{\mathcal{X}_I} \sum_{k\in\mathcal{B}} \left\| \hat{\mathbf{z}}^{b_k}_{b_{k+1}} - \mathbf{H}^{b_k}_{b_{k+1}}\mathcal{X}_I \right\|^2, \tag{18}$$

we can get the velocities in body frame and the gravity vector in the visual base frame $(\cdot)^{c_0}$, as well as the scale parameter. The translational components $\bar{\mathbf{p}}^{c_0}$ from the visual structure will be scaled to the metric units.

3) Gravity Refinement: The gravity vector obtained from the previous step can be refined by constraining the magnitude constraint. In most cases, the magnitude of the gravity vector is known. However, if we directly add this norm constraint into the optimization problem in eq.(18), it will become nonlinear and hard to solve. Here, we use a linear method to enforce this constraint by optimizing the 2D error states on its tangent space. Since the magnitude of gravity is known, the degrees-of-freedom of the gravity is two and we can parameterize the gravity with two variables on its tangent space. We parameterize the gravity as $g\cdot\bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, where g is the magnitude of gravity, $\bar{\hat{\mathbf{g}}}$ is the direction vector of current estimation, $\mathbf{b}_1$ and $\mathbf{b}_2$ are two orthogonal basis spanning the tangent plane. w_1 and w_2 are the corresponding displacements towards $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively, as shown in Fig. 3. We can find one set of $\mathbf{b}_1$, $\mathbf{b}_2$ by cross product easily, as shown in Algorithm 1. Then we substitute $\mathbf{g}$ in eq.17 by $g\cdot\bar{\hat{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$ and it is also in linear form. This process iterates several times until $\hat{\mathbf{g}}$ converges.

After refining gravity vector, we can get the rotation matrix $\mathbf{q}^w_{c_0}$ between world frame and c_0 frame by rotating the gravity to z-axis. According to these rotation matrix, we rotate all variables from frame $(\cdot)^{c_0}$ to the world frame $(\cdot)^w$. The

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

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

velocity in bode frame will also be rotated to world frame. At this point, the initialization procedure is completed and all these metric values will be fed for a tightly-coupled nonlinear visual-inertial estimator.

IV. Tightly-coupled Nonlinear Optimization

After state initialization, we proceed with a sliding window based nonlinear estimator for high-accuracy state estimation. An illustration of the sliding window is shown in Fig. 1. Ceres Solver [33] is used for solving this nonlinear optimization problem.

A. Formulation

The full state vector in the sliding window is defined as (the transpose is ignored for simplicity of representation):

$$\begin{aligned}\mathcal{X} &= \left[\mathbf{x}_0,\, \mathbf{x}_1,\, \cdots \mathbf{x}_n,\, \mathbf{x}_c^b,\, \lambda_0,\, \lambda_1,\, \cdots \lambda_m\right] \\ \mathbf{x}_k &= \left[\mathbf{p}_{b_k}^w,\, \mathbf{v}_{b_k}^w,\, \mathbf{q}_{b_k}^w,\, \mathbf{b}_a,\, \mathbf{b}_g\right], k \in [0, n] \\ \mathbf{x}_c^b &= \left[\mathbf{p}_c^b,\, \mathbf{q}_c^b\right],\end{aligned} \tag{19}$$

where $\mathbf{x}_k$ is the k^{th} frame state, which contains position, velocity, and orientation in the world frame and acceleration bias and gyroscope bias in the body frame. n is the number of keyframes and m is the number of features in the sliding window. λ_l is the inverse depth of the l^{th} feature from its first observation on the unit sphere.

We minimize the sum of prior and the Mahalanobis norm of all measurement residuals to obtain a maximum posteriori estimation:

$$\min_{\mathcal{X}} \left\{ \left\|\mathbf{r}_p - \mathbf{H}_p\mathcal{X}\right\|^2 + \sum_{k\in\mathcal{B}} \left\|\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k},\, \mathcal{X})\right\|_{\mathbf{P}_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j)\in\mathcal{C}} \left\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j},\, \mathcal{X})\right\|_{\mathbf{P}_l^{c_j}}^2 \right\}, \tag{20}$$

where $\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k},\, \mathcal{X})$ and $\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j},\, \mathcal{X})$ are residuals for the IMU and visual models respectively. $\mathcal{B}$ is the set of all IMU measurements, $\mathcal{C}$ is the set of feature which has been observed at least 2 times in the current sliding window. $\{\mathbf{r}_p,\, \mathbf{H}_p\}$ is the prior information from marginalization. Corresponding models are defined in the following sections.

B. IMU Model

Considering two consecutive frames b_k and b_{k+1} in the window, according to the IMU measurement function eq. 12, the residual of IMU measurement model can be defined as:

$$\begin{aligned}\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k},\, \mathcal{X}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\beta}_{b_{k+1}}^{b_k} \\ \delta\boldsymbol{\theta}_{b_{k+1}}^{b_k} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k) - \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \mathbf{R}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w) - \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ 2\left[\mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^w \otimes (\hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k})^{-1}\right]_{xyz} \\ \mathbf{b}_{a b_{k+1}} - \mathbf{b}_{a b_k} \\ \mathbf{b}_{w b_{k+1}} - \mathbf{b}_{w b_k} \end{bmatrix},\end{aligned} \tag{21}$$

where $\left[\cdot\right]_{xyz}$ extracts the vector part of the quaternion $\mathbf{q}$, which is the approximation of error state representation. $[\hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k},\, \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k},\, \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}]^{\mathrm{T}}$ is the pre-integrated IMU measurement using only noisy accelerometer and gyroscope measurements, which is related to accelerometer and gyroscope bias and independent of initial velocity and attitude. $\delta\boldsymbol{\theta}_{b_{k+1}}^{b_k}$ is the minimum error-state representation of quaternion.

C. Vision Model

We define the camera measurement residual on unified unit sphere, which are suitable for small and large FOV camera, such as fisheye and omni-directional cameras. Considering the l^{th} feature which is firstly observed in i^{th} image, the residual for the observation in the j^{th} image is defined as,

$$\begin{aligned}\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j},\, \mathcal{X}) &= \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}^T \cdot (\bar{\mathcal{P}}_l^{c_j} - \frac{\mathcal{P}_l^{c_j}}{\|\mathcal{P}_l^{c_j}\|}) \\ \bar{\mathcal{P}}_l^{c_j} &= {\pi_c}^{-1}(\begin{bmatrix}\hat{u}_l^{c_j} \\ \hat{v}_l^{c_j}\end{bmatrix}) \\ \mathcal{P}_l^{c_j} &= \mathbf{R}_b^c(\mathbf{R}_w^{b_j}(\mathbf{R}_{b_i}^w(\mathbf{R}_c^b\frac{1}{\lambda_l}{\pi_c}^{-1}(\begin{bmatrix}u_l^{c_i} \\ v_l^{c_i}\end{bmatrix}) \\ &\qquad + \mathbf{p}_c^b) + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b),\end{aligned} \tag{22}$$

where $[u_l^{c_i},\, v_l^{c_i}]$ is the first observation of the l^{th} feature that happens in the i^{th} image. $[\hat{u}_l^{c_j},\, \hat{v}_l^{c_j}]$ is the observation of the same feature in the j^{th} image. π_c^{-1} is the back projection model which outputs the unit vector in 3D space. 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 which span the tangent plane of $\bar{\mathcal{P}}_l^{c_j}$, as shown in Fig. 4. We can find one set of $\mathbf{b}_1, \mathbf{b}_2$ easily, as shown in Algorithm 1.

$\mathbf{P}_l^{c_j}$ is the standard covariance of a fixed length in tangent space.

D. Marginalization

In order to bound the computational complexity of optimization-based methods, 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 the marginalized states into a prior.

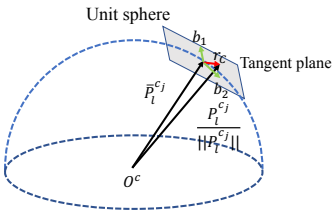


Fig. 4: An illustration of the visual residual on unit sphere. $\bar{\mathcal{P}}_l^{c_j}$ is the ray vector of l^{th} feature observed in the j^{th} frame , $\mathcal{P}_l^{c_j}$ is the transformed ray vector from the i^{th} frame. The residual is defined on the tangent plane of $\bar{\mathcal{P}}_l^{c_j}$.

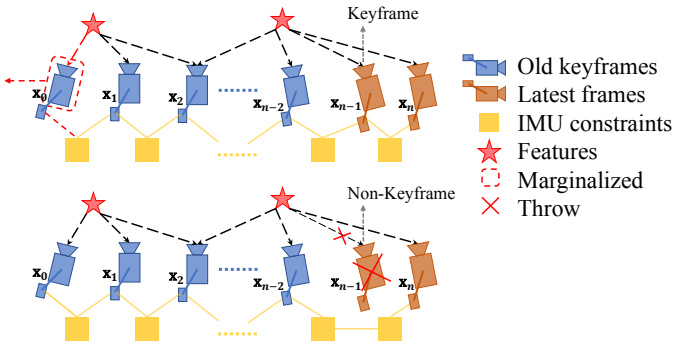


Fig. 5: An illustration of marginalization strategy. If the second latest frame is a keyframe, we will keep it in the window, and marginalize the oldest frame and its corresponding visual and inertial measurements. Marginalized measurements are used to construct a prior. If the second latest frame is non-keyframe, we will throw it and its corresponding visual measurements, and keep the inertial measurements in the window.

As shown in the Fig. 5, when the second latest frame is a keyframe, it will stay in the window and the oldest frame states are marginalized out with its corresponding measurement. Otherwise, if the second latest frame is a non-keyframe, we throw the visual measurements and keep the IMU measurements in the window, instead of marginalizing out all measurements. This strategy will maintain the sparsity of the system. The marginalization scheme can keep spatial keyframes in the window, meanwhile bound the uncertainty for pre-integrated IMU measurements.

We construct a new prior based on marginalized measurements related to the removed state. The marginalization is carried out using the Schur complement. Intuitively, by marginalization, important information of the removed states is kept and computation complexity is bounded.

E. IMU Propagation for High Frequency Outputs

Note that the IMU measurements come at a much higher rate than visual measurements. The frequency of our nonlinear optimization estimator is limited by visual measurements. To benefit the performance for real-time control, the outputs of the estimator are directly propagated with the newest IMU measurements, which serves as the high-frequency feedback in the control loop.

F. Failure Detection and Recovery

Sometimes failure is unavoidable due to violent illumination change or severely aggressive motion. Active failure detection and recovery strategy can improve the practicability of proposed system. Failure detection is an independent module, which detects unpractical outputs of the estimator. We have several criteria for failure detection:

- Tracked feature number of latest frame less than 5;
- A big discontinuity in position or rotation between last two outputs;
- A big value in bias or extrinsic parameters estimation;

If failure is detected, the proposed system switch into initial status, which will try to reinitialize the system. The reinitialize will start at the last possible pose. Meanwhile, the keyframe database will be kept, which will be used for loop closure.

V. Loop Closure

We detect the loop and maintain a pose-only graph in loop thread. Since the sliding window lacks absolute position and yaw observation, whenever a loop closure occurs, the sliding window will attach to the pose graph through relocalization

The keyframe will be added into pose graph after it is margined out from the sliding window. The pose graph will be optimized when the newly added keyframe contains loop information. The IMU measurements render roll and pitch angle fully observable, so the accumulated drift only occurs in four degrees-of-freedom (x, y, z and yaw angle). To avoid importing spurious information, we directly optimize pose graph on these four degrees-of-freedom.

A. Loop Detection

We utilize bag-of-word place recognition which is introduced in DBoW2 [24] with BRIEF descriptors [34] to perform loop detection. When a new keyframe is coming, the number of features extracted in Sect. II-A is not enough for loop detection, we need to extract more features. We extract extra 500 FAST corner features [35]. BRIEF descriptors are used to describe origin observed features and new detected corner features. The descriptors are treated as the visual word to query the visual vocabulary. If loop happens, the DBoW2 will return the loop candidate after temporal and geometrical consistency check. BRIEF descriptors are kept with keyframe, while the raw image is thrown to save memory.

B. Feature Retrieving

When the loop is detected, we need to establish the connection between new keyframe and its loop candidate by retrieved features. We only find connection within observed features extracted in Sect. II-A, since these features have depth information. Feature retrieving is performed by BRIEF descriptor matching. Directly descriptor matching can cause a lot of outliers. In order to reduce outliers, we limit search area within the neighbor of the same position when matching in another frame. Then a geometry consistency check by fundamental matrix test with RANSAC is performed to remove outliers, as shown in Fig. 7. When the number of feature correspondences beyond a certain threshold, we treat this candidate as a right loop detection and fuse its information in the following.

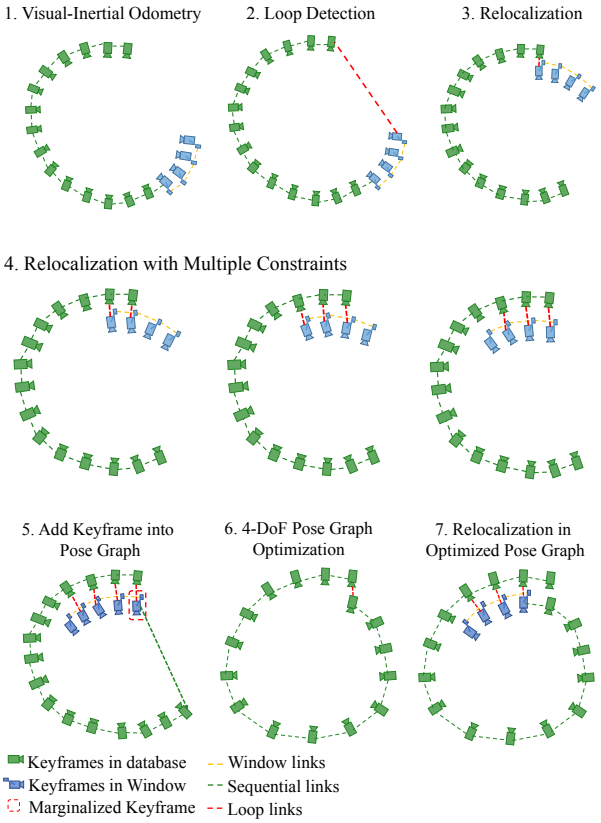
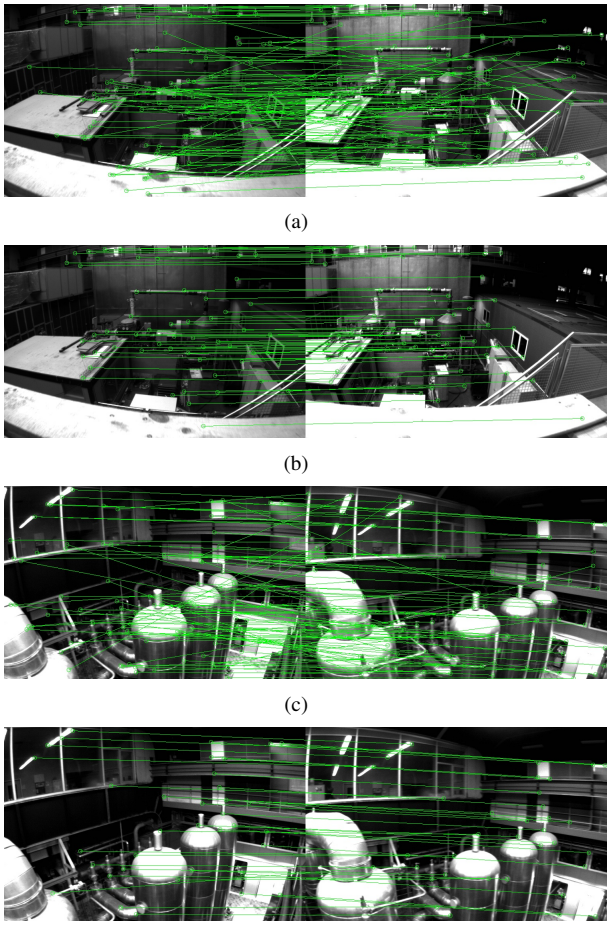


Fig. 6: A timing diagram which illustrates loop clousure procedure. 1 shows the pure visual-inertial odometry. 2 shows a feasible loop is detected for the newest keyframe (Sect. V-A). 3 and 4 show the sliding window is relocalized with multiple loop constraints (Sect. V-C). 5 shows adding keyframe into pose graph when the keyframe is marginlized out of sliding window (Sect. V-D). 6 shows the pose graph is optimized with relative pose constrain (Sect. V-E). 7 shows the sliding window will relocalize after pose graph optimization.



(a) (b) (c) (d)

Fig. 7: (a)(c) Directly BRIEF feature matching results. (b)(d) Results of limited-area BRIEF feature matching and geometric RANSAC rejection. Apparently, outliers are efficiently rejected by limited-area and geometric RANSAC.

C. Relocalization

We use m and v to denote a new keyframe and its looped keyframe. Since new keyframe m will be put into the sliding window, and all frames share the same features in sliding window, we can easily connect looped keyframe v with sliding window by these retrieved features. We jointly optimize the looped keyframe v into the siding window bundle. The looped keyframe is treated as an addition measurement frame with constant pose in sliding window, which only contains several visual constraints without IMU constraint. We can easily write a same visual model for retrieved features in looped frame v as eq. 22. The only difference is that the pose $(\hat{\mathbf{q}}_v^w, \hat{\mathbf{p}}_v^w)$ of looped old frame is fixed, which is obtained from previous odometry.

Adding loop term into whole nonlinear cost function eq. 20:

$$\min_{\mathcal{X}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k\in\mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) \right\|^2_{\mathbf{P}_{b_{k+1}}^{b_k}} + \sum_{(l,j)\in\mathcal{C}} \left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X}) \right\|^2_{\mathbf{P}_l^{c_j}} + \sum_{(l,v)\in\mathcal{L}} \|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}_l^v, \mathcal{X}, \hat{\mathbf{q}}_v^w, \hat{\mathbf{p}}_v^w)\|^2_{\mathbf{P}_l^{c_v}} \right\}, \tag{23}$$

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

D. Adding Keyframe into Pose Graph

When a keyframe is marginalized out from sliding window, it will be added into pose graph. In practice, we add one keyframe into pose graph among every three keyframes to bound computation. This keyframe serves as an vertex in the pose graph, and it will connected other vertexes by two kinds of edges.

1) Sequential edge: one keyframe will establish several sequential edges to its previous keyframes. The physical meaning of sequential edge is the relative transformation between these two frame, which is obtained from pure odometry. Considering the new coming keyframe i and its previous keyframe j, the relative transformation only contains position $\hat{\mathbf{p}}_{ij}^{i}$ and yaw angle $\hat{\psi}_{ij}$ in our 4-DoF pose graph framework:

$$\begin{aligned}\hat{\mathbf{p}}_{ij}^{i} =& \hat{\mathbf{R}}_{i}^{w\,-1}(\hat{\mathbf{p}}_{j}^{w} - \hat{\mathbf{p}}_{i}^{w}) \\ \hat{\psi}_{ij} =& \hat{\psi}_{j} - \hat{\psi}_{i}\end{aligned} \tag{24}$$

2) Loop edge: If the new comming keyframe has detected loop, it will connect the looped frame by the loop edge. Similarly, the loop edge only contains relative position and yaw angle between these two frame. The caculation is same with eq. 24.

E. 4-DoF Pose Graph Optimization

Since the absolute scale, roll and pitch angle are fully observable in the visual-inertial system, accumulate drifts only occur in position (x, y, z) and yaw angle. To take the advantage of observability characteristics and avoid spurious information, we only optimize 4-DoF pose graph when loop detection occurs instead of 6 or 7-DoF optimization.

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

$$\mathbf{r}_{i,j}(\mathbf{p}_{i}^{w}, \psi_{i}, \mathbf{p}_{j}^{w}, \psi_{j}) = \begin{bmatrix} \mathbf{R}(\hat{\phi}_{i}, \hat{\theta}_{i}, \psi_{i})^{-1}(\mathbf{p}_{j}^{w} - \mathbf{p}_{i}^{w}) - \hat{\mathbf{p}}_{ij}^{i} \\ \psi_{j} - \psi_{i} - \hat{\psi}_{ij} \end{bmatrix}, \tag{25}$$

where $\hat{\phi}_{i}, \hat{\theta}_{i}$ are the estimation of roll and pitch angle, obtained from pure odometry, which is fixed in the pose graph optimization. The first row is relative position error, and the second row is relative yaw angle error.

The whole graph of sequential edges and loop edges is optimized by:

$$\min_{\mathbf{p},\psi} \left\{ \sum_{(i,j)\in\mathcal{S}} \|\mathbf{r}_{i,j}\|^{2} + \sum_{(i,j)\in\mathcal{L}} h(\|\mathbf{r}_{i,j}\|) \right\}, \tag{26}$$

where $\mathcal{S}$ is the set of all sequential edges and $\mathcal{L}$ is all loop edges. $h(\cdot)$ is huber robust cost function. In the optimization procedure, we only fix the pose of first keyframe as zero ($\mathbf{p}_{0}^{w} = \mathbf{0}, \phi_{0} = 0$). The reason we use huber cost function for loop constraints is that sometimes false loop detection occurs. Huber cost function can relieve the influence of potential false loop detection. For sequential edges, we don't have such worry since they are extracted from incremental odometry without outliers.

After pose graph optimization, the slide window will be relocalized to the pose graph again.

F. Database Management

Along with the increase of trajectory, the database becomes larger and larger. Loop detection and pose optimization graph time become longer and longer. Although we take the minimum storage strategy (only save pose and features descriptor without raw image), the calculation time will limit real-time performance when working several hours. To this end, we maintain the database in a limited size. We downsample the database according to distribution density when the number of keyframes beyond the certain threshold. We remove keyframes in angle and pose centralized area, and remain the frames which have a minimum distance or angle difference with its neighbor.

VI. Experimental Results

We perform three experiments and two applications to evaluate proposed algorithm. In the first experiment, we compare the proposed algorithm with other state-of-art algorithm on public datasets. We do the numerical analysis to show the accuracy of our system. Secondly, we test our system in the indoor environment to evaluate the performance in repeated scenes. Then a large-scale experiment is carried out to illustrate the long-time practicability of our system. Additionally, we apply proposed system into two devices. The one is the MAV platform. We use VINS as position feedback to control the MAV following designed trajectory. The other one is mobile phone platform. We transplant our implementation on the iOS mobile device and compare with other professional commercial device.

A. Dataset Comparison

We evaluate proposed method with EuRoC MAV Visual-Inertial Datasets [36]. The datasets are collected onboard a micro aerial vehicle, which contain stereo images (Aptina MT9V034 global shutter, WVGA monochrome, 20 FPS), synchronized IMU measurements (ADIS16448, angular rate, and acceleration, 200 Hz), and ground truth states (VICON and Leica MS50). We only use left camera from stereo images set. Nontrivial IMU bias and illumination change are included in the datasets, which make it representative and challenging.

In these experiments, we compare proposed method with OKVIS [11], which is the state-of-the-art visual-inertial odometry working with monocular and stereo cameras. OKVIS is another optimization-based sliding-window algorithm. Our algorithm is different with OKVIS in every detail, and our system is more complete with robust initialization and loop closure. We choose two sequences, MH_03_median, MH_05_difficult to show the performance of proposed method. For simplifying notation, we use VINS to denote our pure odometry and VINS_loop to denote the refined pose graph after loop detection. We use OKVIS_mono and OKVIS_stereo to denote the OKVIS's result working with monocular image and stereo image respectively. To fairly compare the results from two algorithms, for each result, we throw the first 100 output and use the following 150 outputs to calculate the transformation towards the ground truth and align all outputs to the ground truth through this transformation, because the monocular system needs longer time to converge.

For the sequence MH_03_median, the trajectory is shown in Fig. 8. Since little rotation movement occurs in this sequences, we mainly compare translation error. The x, y, z error along

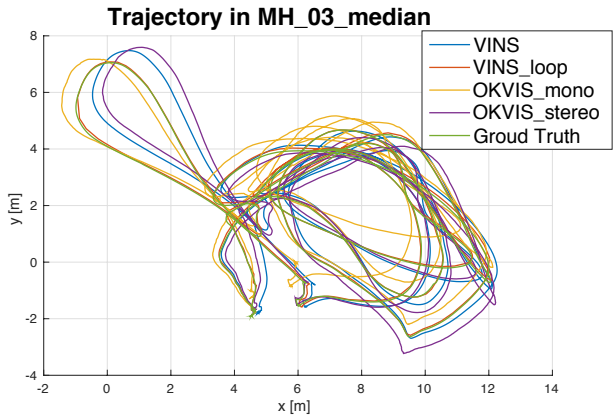


Fig. 8: Trajectory in MH_03_median, compared with OKVIS.

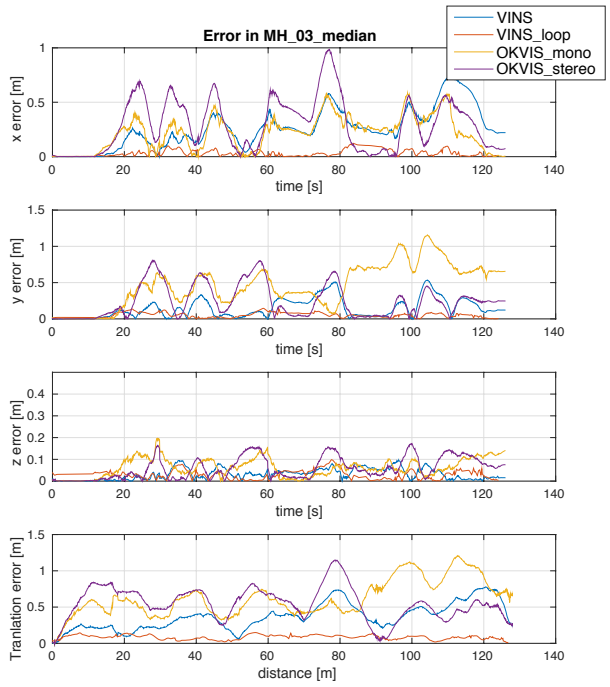


Fig. 9: Translation error plot in MH_03_median.

with time and the translation error along with distance is shown in Fig. 9. In the error plot, the proposed method with loop function has the smallest translation error. The result is same in MH_05_difficult. The proposed method with loop function has the smallest translation error. The translation and rotation error is shown in Fig. 11. Since the movement is smooth without much yaw angle change in this sequence, only position drift occurs in this dataset. Obviously, the loop closure function efficiently bound the accumulated drifts. For the rotation error, OKVIS seems more stable. Honestly, OKVIS performs better in roll and pitch angle estimation. The possible reason may be that proposed method use the pre-integration technique which is the first-order approximation of IMU propagation to save computation resource. Another possible reason is that OKVIS marginalize IMU constraints in a higher frequency.

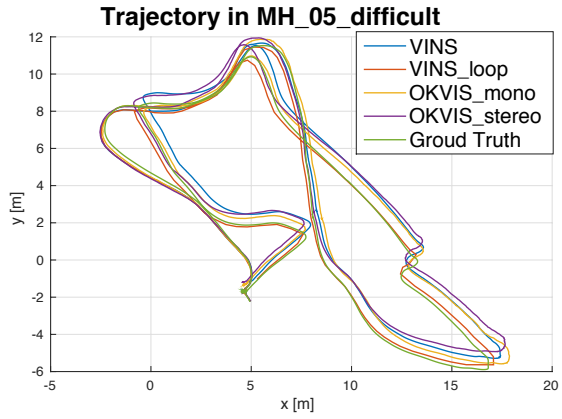


Fig. 10: Trajectory in MH_05_difficult, compared with OKVIS..

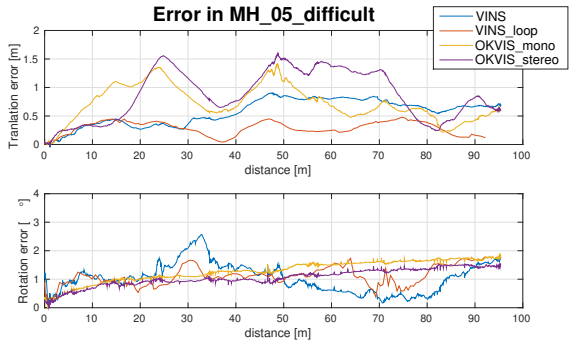


Fig. 11: Translation error and rotation error plot in MH_05_difficult.

Our proposed method performs well in all EuRoC datasets, even in the most challenging sequence, V1_03_difficult, the one includes aggressive motion, texture-less area, and great illumination change. Our proposed method can initialize quickly in V1_03_difficult, due to the robust initialization procedure. The robust initialization procedure can improve the success rate in practice.

Admittedly, for pure odometry, both proposed method and OKVIS are much accurate, it is hard to distinguish which one is better. It is impossible to further increase the accuracy by orders of magnitude for tightly-coupled optimization-based fusion. Our proposed method is a complete system compared with OKVIS since we have robust initialization and loop closure function to assist pure odometry.

B. Indoor Experiment

In the indoor experiment, we choose our laboratory environment as the experiment area. The sensor set we use is shown in Fig. 12, a monocular camera (20Hz) and IMU (100Hz) inside DJI A3 controller[3]. We hold the sensor set by hand and walk in the laboratory around and around. We encounter pedestrians, darkness, low-texture area, glass and reflection, as shown in Fig. 13, which represents the normal daily life. Details can be found in the supplementary video.

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

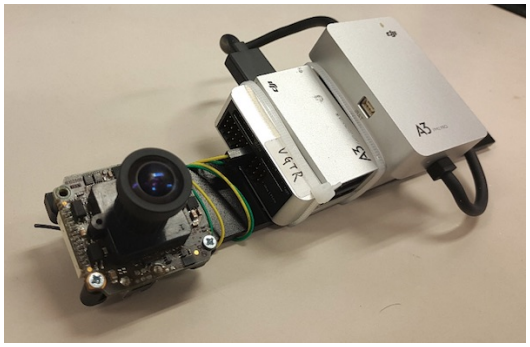
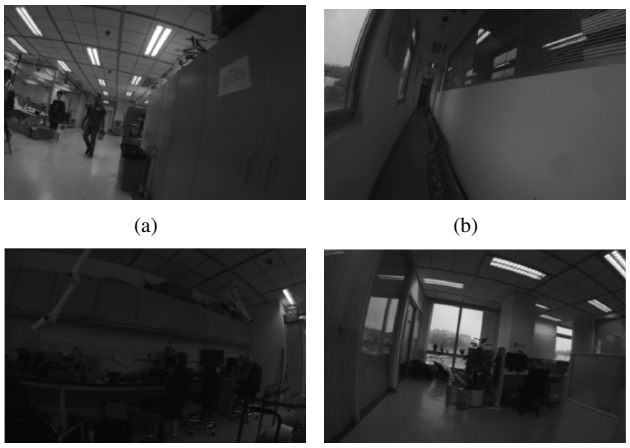


Fig. 12: The device used in the indoor experiments. One forward-looking global shutter camera (MatrixVision mvBlueFOX-MLC200w) with 752×480 resolution. Inertial measurement unit (IMU, ADXL278 and ADXRS290, 100Hz) inside DJI A3 flight controller.



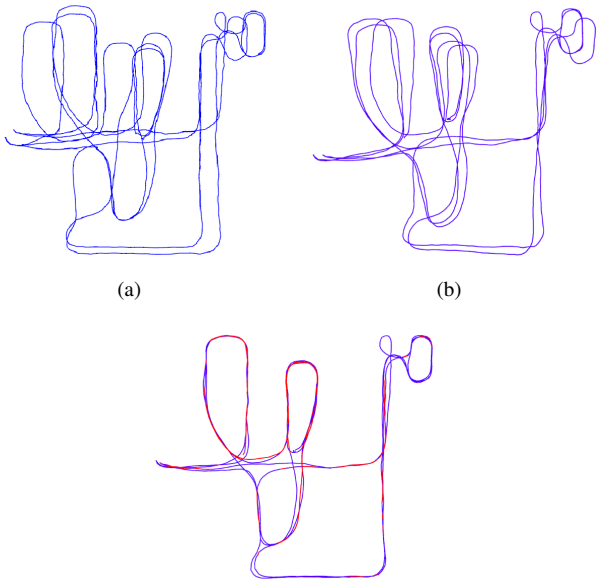
(a) (b) (c) (d)

Fig. 13: (a) Pedestrians. (b) Low-texture area. (c) Darkness. (4) Glass and reflection.

We compare our result with OKVIS, as shown in Fig. 14. The Fig. 14(a) is the odometry from OKVIS. Fig. 14(b) is the pure odometry from proposed method without loop closure. Fig. 14(c) is the result of proposed method with loop closure. Apparently, nontrivial drifts occur when we walk indoor around and around. Both OKVIS and proposed pure odometry accumulate great drifts in x, y, z, and yaw angle. Our loop closure function can correct drifts efficiently.

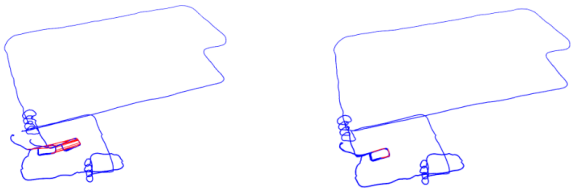
C. Large-scale Environment

1) Go out of lab: We test proposed algorithm from the indoor environment to outdoor environment. The sensor that we use is the same with that used in the indoor environment, a Bluefox camera with IMU inside A3. We start from the seat in the laboratory and go around the laboratory. Then we go down the stairs and walk around the playground outside the building. Next, we go back to the building and go upstairs. Finally, we return to the seat in the laboratory. The whole trajectory is more than 700 meters and last ten minutes. The details can be found in the supplementary video.



(a) (b) (c)

Fig. 14: (a) Trajectory of OKVIS. (b) Trajectory of proposed method without loop closure. (b) Trajectory of propose method with loop closure. The red path is the places where loop are detected.



(a) (b)

Fig. 15: Trajectory of going out of laboratory. (a) The trajectory without loop closure. (b) The trajectory with loop closure. The red lines are loop links. The spiral lines are going up and down stairs.

The trajectory is shown in Fig. 15. The trajectory without loop closure is shown in Fig. 15(a), while the trajectory with loop closure is shown in Fig. 15(b). The trajectory is aligned with Google Map in Fig. 16.

Without loop closure, the distance between the start point and end point is [-5.47, 2.76. -0.29] in x, y and z axis, which occupies 0.88% in the whole length. With loop correction. the distance between the start point and end point is [-0.032, 0.09, -0.07], which is trivial compared with the whole length. In addition, the optimized trajectory is smooth, which can be precisely aligned with the satellite map.

2) Go around campus: This large-scale experiment was recorded with the handheld VI-Sensor[4] walking around HKUST campus, which is around 710m in length, 240m in

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

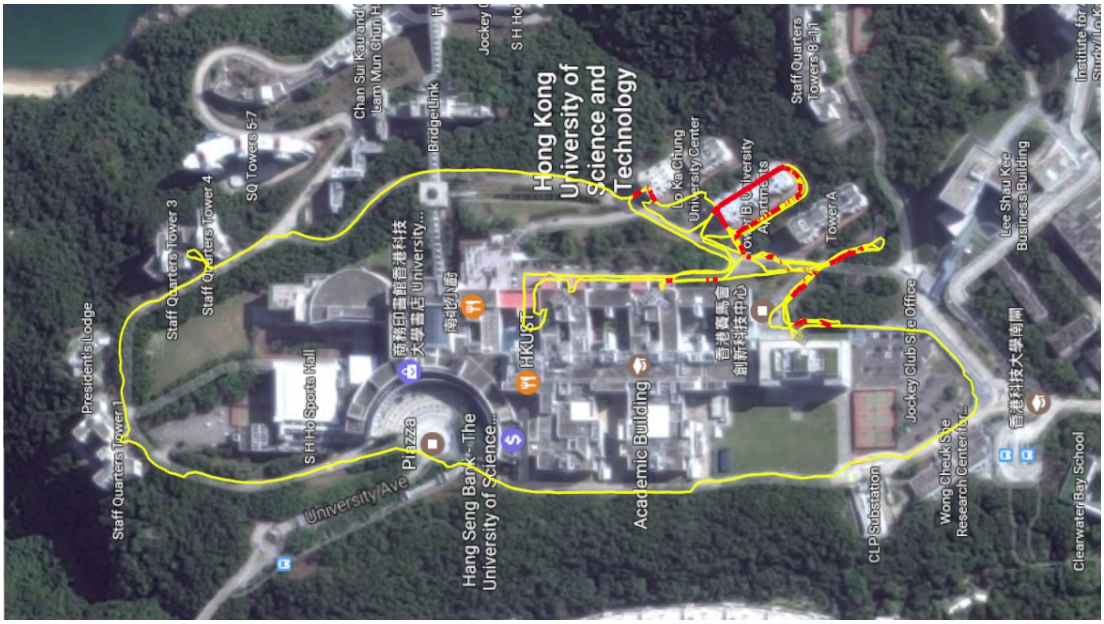


Fig. 17: The trajectory of large-scale environment aligned with Google map. The yellow line is the trajectory of proposed method. The red line is loop detected area.

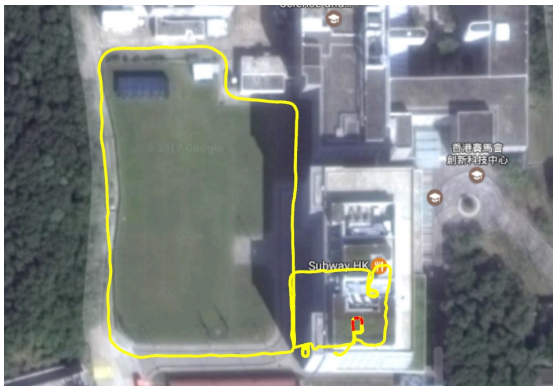


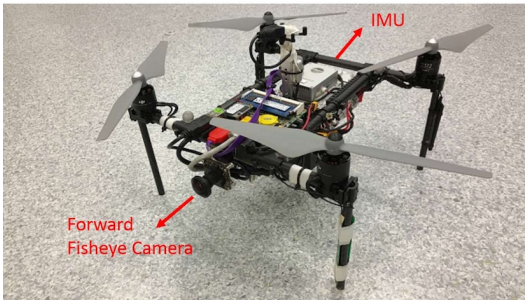
Fig. 16: The trajectory of going out of lab aligned with Google Map. The yellow line is the trajectory of proposed method. The red line is loop detected area.

width and 60m in height. The whole path length is 5.62km. The data contains the 25Hz image and 200Hz IMU lasting for 1 hour and 34 minutes. It's a good long-time experiment to test the stability and durability of the system.

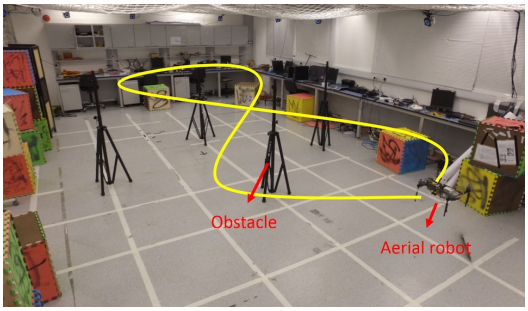
In this large-scale test, We set the keyframe database size as 2000, which can provide sufficient loop information and achieve real-time performance. Every time when the loop is detected, we optimize the whole pose graph once. We run this test with Intel i7-4790 CPU, 3.60GHz. The timing statistics is show in Table.I. The trajectory is aligned with Google map in Fig.17. Compared with Google map, we can see our results are almost drift-free in such a long run.

D. Application I: Onboard Aerial Robot

We apply our algorithm onboard an aerial robot, as shown in Fig. 18(a). One forward-looking global shutter camera (MatrixVision mvBlueFOX-MLC200w) with 752×480 resolution.



(a)



(b)

Fig. 18: (a) The self-developed aerial robot with one forward-looking fisheye camera (MatrixVision mvBlueFOX-MLC200w, 190 FOV) and IMU (ADXL278 and ADXRS290 inside in DJI A3, 100Hz). (b) The designed trajectory in onboard experiment. Four position-known obstacles are put in the test ground. The yellow line is the design eight-figure trajectory which the aerial robot should follow. The robot follows the trajectory four times without loop closure. Details are shown in the supplementary video.

TABLE I: Timing Statistics

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

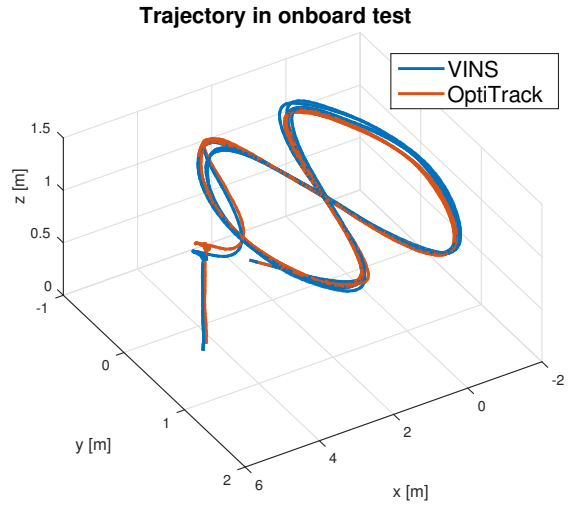


Fig. 19: The trajectory of VINS on the MAV platform without loop closure compared with ground truth. The MAV follows the trajectory four times. VINS outputs are used as real-time position feedback in the real-time control loop. The ground truth is provided by OptiTrack. Total length is $61.97m$. Final drift is $0.18m$.

It is equipped with a 190-degree fisheye lens. A DJI A3 flight controller is used both as the inertial measurement unit (IMU, ADXL278 and ADXRS290, 100Hz) and attitude stabilization control. The onboard computation resource is an Intel i7-5500U CPU running at 3.00 GHz. Tradition pinhole camera model is not suitable for large FOV camera. We use MEI [37] model for this camera, calibrated by the toolkit introduced in [38].

In this experiment, we test the performance of autonomous trajectory tracking under the VINS estimation. No loop closure is used in 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. 18(b). The four obstacles are put around the trajectory to verify the accuracy of monocular VINS without loop closure. The quadrotor follows this trajectory four times continuously during the experiment. The 100 Hz state estimation results achieve the real-time feedback to control the quadrotor.

The robustness and accuracy are of vital importance to this real-time onboard experiment. The final drift is [0.08, 0.09, 0.13] m, relative to the total 61.97 m path length, by comparing with OptiTrack[5] without loop closure. The final percentage of drift is 0.29%. The details of the translation and rotation as well as their corresponding error are shown in Fig. 20.

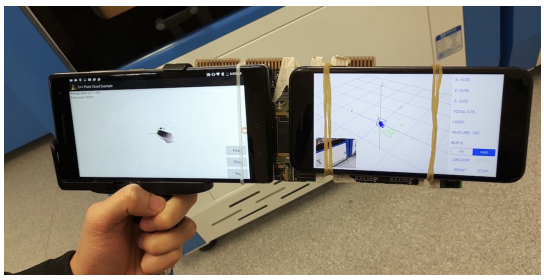


Fig. 21: The simple holder that we used to mount the Tango device (left) and the iPhone7 Plus (right) on which our algorithm runs.

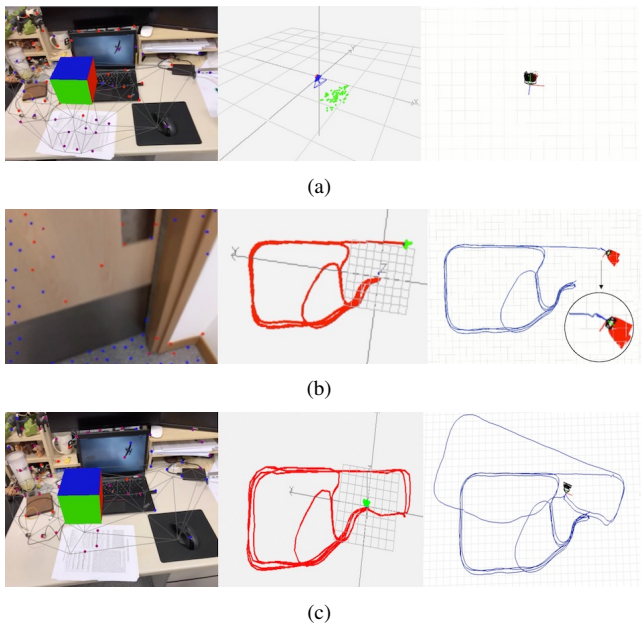


Fig. 22: From left to right: The AR image of VINS. The trajectory of VINS. The trajectory of Tango device. (a) The VINS and Tango are initialized at the start location and a virtual box is inserted on the plane which extracted from estimated features. (b) A challenging case in which the cameras watch the moving door. The drift of Tango trajectory is highlighted. (c) The whole trajectories of VINS and Tango during we walk inside and outside the room. The total length is about 264m.

E. Application II: Mobile Device

To validate the practicability of the proposed algorithm, we port the whole system to mobile devices and present a simple AR application to show its accuracy and robustness. We compare the performance of VINS with Google Tango device[6], which is one of the best commercial augmented reality solutions on mobile platforms in the current stage.

In this experiment, we implement the whole VINS system on iPhone7 Plus. we use 30 Hz images with 640 $\times$ 480

[5]http://optitrack.com/

[6]http://shopap.lenovo.com/hk/en/tango/

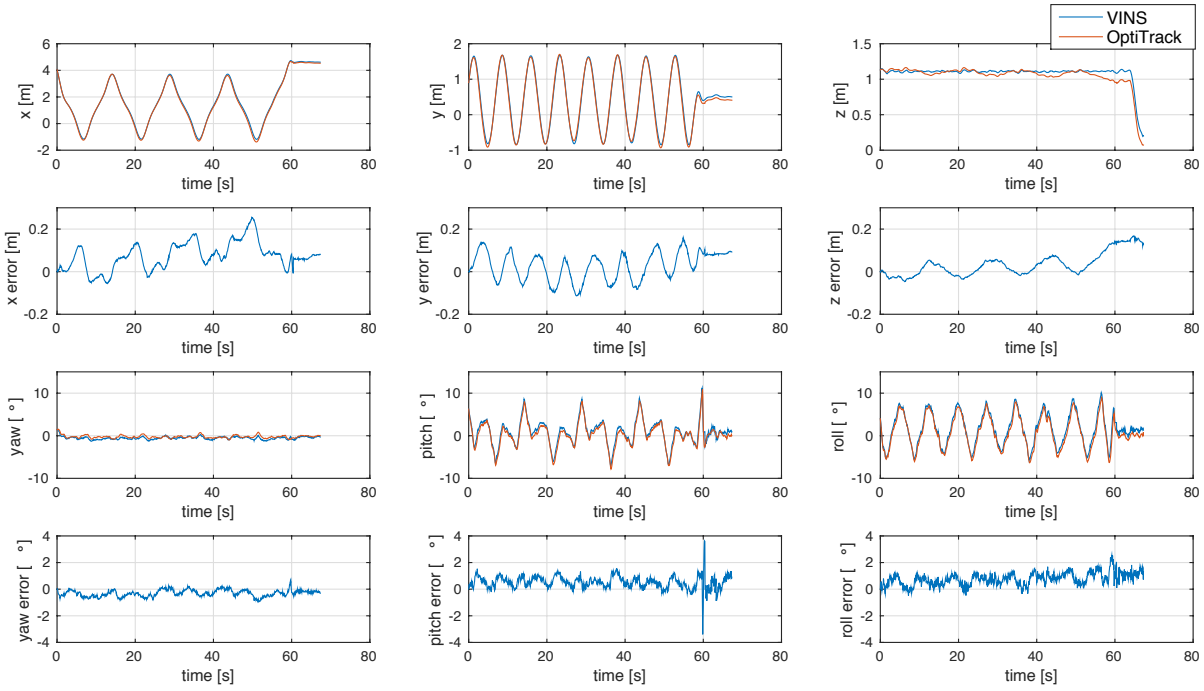


Fig. 20: Position, orientation and their corresponding error of proposed system compared with OptiTrack in the onboard experiment. An impulse in pitch error at the 60s is caused by a violent break at the end of designed trajectory.

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. 21 shows, we mount the iPhone with the Tango-Enabled smartphone Lenovo Phab 2 Pro which uses the global shutter fisheye camera, synchronized IMU, to estimate the state and percept the environment. Firstly, we insert a virtual cube on the plane which is extracted from estimated visual features as shown in Fig. 22(a). Then we hold the two devices and walk inside and outside the room in a normal pace. When the loop is detected, we use the 4-DoF pose graph optimization (Sect. V-E) to correct the x, y, z and yaw drift for all the keyframes between the looped old frame and the current frame in keyframe database. When we are opening a door, Tango's yaw estimation jumps a big angle, as shown in Fig. 22(b). The reason maybe estimator crash caused by unstable feature tracking or active failure detection and recovery. Our system still works well in this challenging case. After traveling about 264m, we return to the start location. The whole result can be seen in Fig. 22(c), the trajectory of tango occurs drift in the last lap while our VINS returns to the start point and 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 proposed estimation. However, this experiment shows that the proposed method can run on general mobile devices and have the potential ability to compare with the very professional devices. The robustness of proposed method is also proved in this experiment. The experiment details can be found in the supplementary video.

VII. Conclusion

In this paper, we perform a complete monocular visual-inertial system for 6-DoF state estimation. Our system is robust to initialization with unknown states, online camera-IMU extrinsic parameter calibration, loop detection and 4-DoF pose graph optimization. These characters make our system practical and easy-to-use. Two applications prove that our system has a great potentiality to be extended for other platforms.

In the future, we will do the dense mapping based on the robust state estimation and sparse features from VINS.

Acknowledgment

Thanks Lin Yonggen for large-scale environment dataset collection.

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] 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, 2014.

[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] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Y. Ling, T. Liu, and S. Shen, "Aggressive quadrotor flight using dense visual-inertial fusion," in *Proc. of the IEEE Int. Conf. on Robot. and Autom.* IEEE, 2016, pp. 1499–1506.

[13] 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.

[14] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Int. Sym. of Robot. Research*, Flagstaff, AZ, Aug. 2011.

[15] 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.

[16] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. Robot. Research*, vol. 30, no. 1, pp. 56–79, Jan. 2011.

[17] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics (TRO)*, 2016.

[29] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

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

[31] 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.

[32] 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.

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

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

[35] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer vision–ECCV 2006*, pp. 430–443, 2006.

[36] 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.

[37] 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.

[38] 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.