sensors

MDPI

Article

Monocular Visual-Inertial SLAM: Continuous Preintegration and Reliable Initialization

Yi Liu [1], Zhong Chen [1,*], Wenjuan Zheng [2], Hao Wang [2] and Jianguo Liu [1]

[1] National Key Laboratory of Science and Technology on Multi-Spectral Information Processing, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China; skyridermike@hust.edu.cn (Y.L.); jgliu@hust.edu.cn (J.L.)

[2] Beijing Aerospace Automatic Control Institute, Beijing 100854, China; dingling721@126.com (W.Z.); whipraihust@163.com (H.W.)

* Correspondence: henpacked@hust.edu.cn; Tel.: +86-27-8755-7746

Received: 5 September 2017; Accepted: 6 November 2017; Published: 14 November 2017

Abstract: In this paper, we propose a new visual-inertial Simultaneous Localization and Mapping (SLAM) algorithm. With the tightly coupled sensor fusion of a global shutter monocular camera and a low-cost Inertial Measurement Unit (IMU), this algorithm is able to achieve robust and real-time estimates of the sensor poses in unknown environment. To address the real-time visual-inertial fusion problem, we present a parallel framework with a novel IMU initialization method. Our algorithm also benefits from the novel IMU factor, the continuous preintegration method, the vision factor of directional error, the separability trick and the robust initialization criterion which can efficiently output reliable estimates in real-time on modern Central Processing Unit (CPU). Tremendous experiments also validate the proposed algorithm and prove it is comparable to the state-of-art method.

Keywords: sensor fusion; SLAM; computer vision; inertial navigation; tightly coupled

1. Introduction

Simultaneous Localization and Mapping (SLAM) has attracted a lot of attention from both robotic community and industrial community. Laser scanner was the primary sensor in earlier SLAM works (e.g., [1,2]). However, the size and weight of laser scanner significantly constrain the agility of the platform and thus the use of vision sensor gradually became a tendency [3–8]. The advantages of vision sensor include cheaper price, lighter weight and lower power consumption, which are essential to mobile platforms (e.g., Micro Aerial Vehicle). Furthermore, vision sensor has the capability for retrieving the environment's appearance, color and texture such that it is possible to perform some high-level tasks such as scene recognition. While stereo camera [9–11], RGB-D camera [12,13] and omnidirectional camera sensors [14] have been proven suitable in some certain scenarios and applications, monocular SLAM provides a fundamental solution.

A typical SLAM system is composed of front-end and back-end. Front-end is in charge of performing data association for the back-end module. For visual SLAM, feature-based method and direct method are two main approaches in the front-end module. Direct method (e.g., [15–17]) directly uses the intensity values in the image to estimates the structure and motion, showing more robust than feature-based method in the texture-less scenarios. However, feature-based method is much less sensitive to exposure adjustment in video/image, and it may be a better choice for robust tracking under rich texture environment due to the invariance of descriptor. Back-end in SLAM is in charge of state inference after data association. From the viewpoint of the probabilistic framework, the purpose of back-end is to output the MAP (Maximum a posterior) estimates given the measurements from front-end. For this purpose, the back-end solutions have evolved from filter

based approaches [3,18–22] to graph optimization methods [7,8,23]. The first real-time monocular SLAM system was presented by Davsion [3] with Extended Kalman Filter (EKF) framework, and Civera [18] improved its performance with the inverse depth feature parametrization. However, the maintaining of the dense covariance matrix in EKF is very expensive so that the size of features has to be very limited. Compared to filter-based methods, Graph optimization exploits the sparse structure and thus it enables fast computation by using sparse linear solvers. Current optimization solvers (e.g., g2o [24], Ceres [25], iSAM [26], GTSAM [27]) are able to solve a typical optimization problems with tens thousands of variables in few seconds. There also exists different strategies for combing the front-end and back-end. Klein [7] presented a novel parallel system. This real-time system consists of the tracking thread and the mapping thread. Motivated by the parallel design, Raul [23] presented an improved system with the concept of co-visibility graph for local mapping to efficiently keep the consistency for large scale environment. Forster [15] also utilized a parallel system by combining direct tracking for pose estimation and depth filter for feature estimation. Besides these methods, the sliding window strategy also shows good performance [11,28–30] and it keeps the computational time bounded by marginalizing out old states.

On the other hand, inertial measurement unit (IMU), as a complementary sensor to camera, is gradually used in the field of SLAM because it allows the recovery of the global roll, pitch and the undetermined scale in monocular SLAM. The early works of visual-inertial fusion were loosely coupled approaches [21,30–32] and then tightly-coupled approaches proved its superior performance that jointly optimize all state variables [11,33–35]. Among these tight fusion approaches mentioned above [11,20,29] are feature based approaches, which require the feature points that present a high degree of saliency. Mourikis [19] provided MSCKF algorithm and then consistency analysis of MSCKF was followed by [35,36], and [11,28,29] performed optimization framework by a sliding window to limit the computation. Forster [33] proposed the IMU preintegration on a manifold for sensor fusion and the iSAM back-end for incremental optimization. In contrast to these feature-based approaches, direct method fusion with inertial sensor provided by Alejo Concha [34] is the first work that combines the direct method with inertial fusion. Although direct methods are able to track features very efficiently, but they are more likely to fail due to exposure adjustment in vision camera sensor. From the viewpoint of computational complexity, the approaches based on sliding window like MSCKF can be thought as a constant-time solution for each visual frame, but they suffer from relatively larger drift since (a) the commonly used marginalization step usually leads to inconsistent estimates because the invariance is obeyed [35,37]; (b) the earlier observations are neglected. The work in [33] is done by an incremental optimization strategy (iSAM), but one of disadvantage is the unbounded complexity of memory, which can grow continuously over time.

In this paper, we present a visual-inertial navigation system (VINS) that combines the visual SLAM approach and IMU preintegration technique [33,38] beyond the framework of ORB-SLAM [23] and PTAM [7]. Firstly, we derive a new IMU factor, motivated by the work in [35] with the corresponding preintegration method. The derivation is based on the continuous form which allows the use of high-order integration like Runge-Kutta. We stress that the derived IMU factor does not depend on the assumption that the IMU biases keep unchanged between two consequential keyframes such that our proposed IMU factor can better capture the correlation of state uncertainties. Thanks to the proposed IMU factor, given IMU poses (up to a scale) and the preintegrated measurements, we derive a linear least square formulation to initialize the system, which does not need to separately estimate the state variables. More important, since the proposed initialization method has considered the propagated uncertainty and the magnitude of the gravitational vector, we can have a robust mechanism to decide whether current information for initialization is enough or not, which is beyond the discuss in [38–40]. We then propose a well-designed parallel framework Figure 1 that runs a tracking thread and a local mapping thread at the same time. In the tracking thread, we only optimize the current IMU state with the IMU preintegration technique and the current vision factor for low computation cost. In the local mapping thread, we optimize all IMU states in the co-visibility graph

$\mathcal{G}$ and all map points observed in the $\mathcal{G}$ together for a more consist map. For faster convergence, we employ the separability trick in the optimization that subtly uses the overlooked property–IMU velocity and biases are linear in the cost function of the proposed IMU factor.

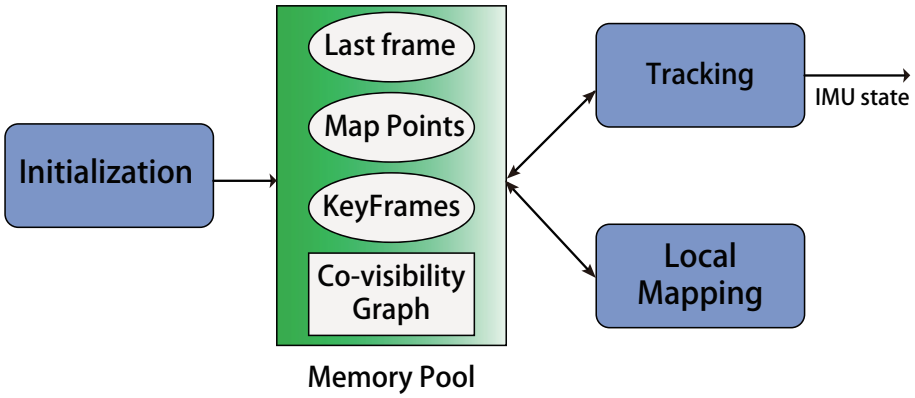


Figure 1. The global framework for our state estimation system. Note the Tracking and Local Mapping are two paralleled threads. A memory pool is utilized during the whole algorithm, which contains the states of map points, keyframes, and last frame. It also maintains the co-visibility graph for both the tracking and local mapping thread. Tracking thread uses the data from the memory pool to produce the state estimation in real-time. Meanwhile, the local mapping thread refines the data in the memory pool, which guarantees the estimation of tracking thread. Two threads perform different tasks and cooperate through the data in the memory pool. The memory pool will be initialized by the initialization process and will be described in Section 4. In the Sections 3.1 and 3.2 we will discuss the tracking thread and the local mapping thread.

The rest of the paper is organized as follows. Section 2 introduces the graph optimization used in estimation and the proposed IMU factor with the corresponding preintegration method. Section 3 presents our work for the tightly coupled approach for visual-inertial SLAM algorithm. Section 4 gives the principle of initialization for our monocular visual-inertial SLAM algorithm. Initialization scheme is by no means trivial for a monocular visual-inertial SLAM because initial feature depth and IMU biases can have significant effects on tightly-coupled SLAM system and the estimator usually suffers from the ill-conditioned cases (e.g., constant velocity). Notations: To simplify the presentation, the vector transpose operators are omitted for the case $\mathbf{A} = \left[\mathbf{a}^\intercal, \mathbf{b}^\intercal, \cdots, \mathbf{c}^\intercal\right]^\intercal$.

2. Graph Optimization

In this section, we adopt the formalism of factor graph [27] and derive a nonlinear least squares formulation to calculate the maximum a posterior (MAP) estimate of the visual-inertial state estimation problem.

2.1. IMU Factor with Preintegration

2.1.1. IMU State and Motion Model

The IMU state to be estimated can be represented by a tuple, i.e.,

$$\mathbf{X} = (\mathbf{R}, \mathbf{p}, \mathbf{v}, \mathbf{b}_g, \mathbf{b}_a) \tag{1}$$

where $\mathbf{b} := (\mathbf{b}_g, \mathbf{b}_a)$, $(\mathbf{R}, \mathbf{p}) \in \mathbb{SE}(3)$ denotes the IMU pose in the global frame, $\mathbf{v} := \dot{\mathbf{p}} \in \mathbb{R}^3$ denotes the IMU velocity expressed in the global frame, $\mathbf{b}_g(t) \in \mathbb{R}^3$ denotes the gyroscope bias and $\mathbf{b}_a(t) \in \mathbb{R}^3$ denotes the accelerometer bias.

An IMU sensor consists of a 3-axis gyroscope and a 3-axis accelerometer. The gyroscope reading at the time t is corrupted by the bias and noise: $\mathbf{w}(t) = \bar{\mathbf{w}}(t) + \mathbf{b}_g(t) + \mathbf{n}_g$, where $\bar{\mathbf{w}}(t)$ denotes the actual IMU angular velocity at the time t, $\mathbf{n}_g$ is assumed to be a white Gaussian noise. Note that the effects form earth rotation is neglected. The accelerometer reading at the time t is also corrupted by the bias and noise: $\mathbf{a}(t) = \mathbf{R}^\intercal(t)(\dot{\mathbf{v}} - \mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a$, where $\mathbf{g}$ is the gravity vector in the global frame and $\mathbf{n}_a$ is also modeled as a white Gaussian noise.

Employing the IMU measurements model above and the random walk model for the time-varying IMU biases, we can easily conclude the IMU motion model in the following:

$$\begin{aligned}\dot{\mathbf{X}} =& f(\mathbf{X}, \mathbf{u}, \mathbf{n}) \\ =& \left(\mathbf{R}S(\mathbf{w} - \mathbf{b}_g - \mathbf{n}_g), \mathbf{R}(\mathbf{a} - \mathbf{b}_a - \mathbf{n}_a) + \mathbf{g}, \mathbf{v}, \mathbf{n}_{bg}, \mathbf{n}_{ba}\right)\end{aligned} \tag{2}$$

where the skew symmetric operator $S(\cdot)$ is given in Appendix, $\mathbf{u} := \begin{bmatrix}\mathbf{w}, \mathbf{a}\end{bmatrix}$ are the measurements from IMU and $\mathbf{n} := \begin{bmatrix}\mathbf{n}_g, \mathbf{n}_a, \mathbf{n}_{bg}, \mathbf{n}_{ba}\end{bmatrix}$ are the white Gaussian noise with the known covariance $\Sigma \in \mathbb{R}^{12\times 12}$.

2.1.2. Mean Propagation

Ignoring the IMU noise $\mathbf{n}$, we have a nominal IMU motion model:

$$\dot{\hat{\mathbf{X}}} = f(\hat{\mathbf{X}}, \mathbf{u}, \mathbf{0}) \tag{3}$$

where $\hat{\mathbf{X}}$ denotes the nominal IMU state. Given the IMU state $\mathbf{X}_i$ and the IMU measurements $\mathbf{u}_{i:j}$ between the time step i and j, the predicted IMU state $\hat{\mathbf{X}}_j$ at the time-step j can be recursively computed via the nominal motion model

$$\hat{\mathbf{X}}_j = F(\mathbf{X}_i, \mathbf{u}_{i:j}) \tag{4}$$

Note that the transformation $F(\cdot, \mathbf{u}_{i:j})$ above represents a series of integral operations and thus a naive implementation of computing $F(\mathbf{X}, \mathbf{u}_{i:j})$ is time-consuming and memory-occupied. Later we will provide a method to efficiently compute $F(\mathbf{X}, \mathbf{u}_{i:j})$ without need to re-integration.

2.1.3. Error-State Motion Model

To concisely quantify the effects of IMU noise, we employ an error between the nominal IMU state $\hat{\mathbf{X}}$ and the actual IMU state $\mathbf{X}$ motivated from [35]

$$\mathbf{e} := \hat{\mathbf{X}} \ominus \mathbf{X} := \begin{bmatrix}\log(\mathbf{R}^\intercal\hat{\mathbf{R}}) \\ \mathbf{R}^\intercal(\hat{\mathbf{v}} - \mathbf{v}) \\ \mathbf{R}^\intercal(\hat{\mathbf{p}} - \mathbf{p}) \\ \mathbf{b}_g - \hat{\mathbf{b}}_g \\ \mathbf{b}_a - \hat{\mathbf{b}}_a\end{bmatrix} \in \mathbb{R}^{15} \tag{5}$$

Based on the nominal motion Equation (2) and the actual motion Equation (3) and the error Equation (5), we now can obtain the error-state propagation model:

$$\dot{\mathbf{e}} \approx \mathbf{A}\mathbf{e} + \mathbf{B}\mathbf{n} \tag{6}$$

where

$$\mathbf{A} = \begin{bmatrix}-S(\hat{\mathbf{w}}) & \mathbf{0} & \mathbf{0} & -\mathbf{I}_3 & \mathbf{0} \\ -S(\hat{\mathbf{a}}) & -S(\hat{\mathbf{w}}) & \mathbf{0} & \mathbf{0} & -\mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 & -S(\hat{\mathbf{w}}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \tag{7}$$

and

$$\mathbf{B} = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix}. \tag{8}$$

Note that the error-state propagation Equation (2) is almost an autonomous linear system, independent of state $\mathbf{x}$. Therefore,

- The covariance $\mathbf{P}$ of $\mathbf{e}$ can be accurately computed by using the following differential equation:

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^\intercal + \mathbf{B}\Sigma\mathbf{B}^\intercal \tag{9}$$

- The autonomous linear system can guarantee safe and reliable preintegration in the sense of the first-order approximation. Given $F(\mathbf{X}_i, \mathbf{u}_{i:j})$, we can easily calculate $F(\mathbf{X}, \mathbf{u}_{i:j})$ based on the linear system theory for any $\mathbf{X}$ as the following:

$$F(\mathbf{X}, \mathbf{u}_{i:j}) = F(\mathbf{X}_i, \mathbf{u}_{i:j}) \oplus \mathbb{A}(\mathbf{X} \ominus \mathbf{X}_i) \tag{10}$$

  where $\oplus$ is the inverse of the operation $\ominus$ defined in (5):

$$\mathbf{X} \oplus \mathbf{e} = (\mathbf{R}\exp(\mathbf{e}_1), \mathbf{v} + \mathbf{R}\mathbf{e}_2, \mathbf{p} + \mathbf{R}\mathbf{e}_3, \mathbf{b}_g + \mathbf{e}_4, \mathbf{b}_a + \mathbf{e}_5) \tag{11}$$

  The matrix $\mathbb{A} \in \mathbb{R}^{15\times 15}$ can be pre-integrated from the following differential equation

$$\dot{\mathbb{A}} = \mathbf{A}\mathbb{A} \tag{12}$$

  with the initial state $\mathbb{A}(t_i) = \mathbf{I}$. Here we stress that (10) makes hundreds of measurements $\mathbf{u}_{i:j}$ unnecessary to be stored after preintegration.

The matrix $\mathbb{A}$ in (12) contains 225 elements. Fortunately, we can simplify the expression of $\mathbb{A}$ as the following:

$$\mathbb{A} = \begin{bmatrix} \mathbf{J}_1^\intercal & \mathbf{0} & \mathbf{0} & \mathbf{J}_4 & \mathbf{0} \\ -\mathbf{J}_1^\intercal S(\mathbf{J}_2) & \mathbf{J}_1^\intercal & \mathbf{0} & \mathbf{J}_5 & \mathbf{J}_4 \\ -\mathbf{J}_1^\intercal S(\mathbf{J}_3) & -\Delta t\mathbf{J}_1^\intercal & \mathbf{J}_1^\intercal & \mathbf{J}_6 & \mathbf{J}_7 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \tag{13}$$

where $\mathbf{J}_1 \in \mathbb{SO}(3)$, $\mathbf{J}_2 \in \mathbb{R}^3$, $\mathbf{J}_3 \in \mathbb{R}^3$, $\mathbf{J}_i \in \mathbb{R}^{3\times 3}$ $(i = 4, 5, 6, 7)$ can be preintegrated by the following differential equation:

$$\begin{aligned} &\dot{\mathbf{J}}_1 = \mathbf{J}_1 S(\hat{\mathbf{w}}),\ \dot{\mathbf{J}}_2 = \mathbf{J}_1 S(\hat{\mathbf{a}}) \\ &\dot{\mathbf{J}}_3 = \mathbf{J}_2,\ \dot{\mathbf{J}}_4 = -S(\hat{\mathbf{w}}_t)\mathbf{J}_4 - \mathbf{I}_3 \\ &\dot{\mathbf{J}}_5 = -S(\hat{\mathbf{a}})\mathbf{J}_4 - S(\hat{\mathbf{w}})\mathbf{J}_5,\ \dot{\mathbf{J}}_6 = \mathbf{J}_5 - S(\hat{\mathbf{w}})\mathbf{J}_6 \\ &\dot{\mathbf{J}}_7 = \mathbf{J}_4 - S(\hat{\mathbf{w}})\mathbf{J}_7 \end{aligned} \tag{14}$$

with the initial guess $\mathbf{J}_1(0) = \mathbf{I}_3$ and $\mathbf{J}_i(0) = \mathbf{0}$ $(i = 2, \cdots, 7)$.

Remark 1. *Compared to the methods proposed in Forster [33] and HKST [30], our derivation is more straightforward and simple. Firstly, our proposed preintegration is born to be continuous. Secondly, unlike the preintegration of Forster and HKST, both of them fix the bias first when computing the 3 preintegration factors and simply use the first order Tyler expansion for the approximate Jacobian of IMU bias and IMU factor, our proposed IMU preintegration factor is based on the entire IMU state(including bias), use continuous*

differential equations which can better capture the correlations inside the IMU state. Thirdly, the defined error is invariant under the yaw angle transformation.

2.1.4. IMU Factor

Given the IMU state $\mathbf{X}_i$ and the IMU measurements $\mathbf{u}_{i:j}$ between the time-step i and j, we have the predicted state $\hat{\mathbf{X}}_j$ as presented in (4). According to the error definition (5) and the error-state motion model (6), we can get the uncertainty between the predicted state $\hat{\mathbf{X}}_j$ and the actual state $\mathbf{X}_j$

$$\hat{\mathbf{X}}_j \ominus \mathbf{X}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{ij}) \tag{15}$$

where the covariance matrix $\mathbf{P}_{ij}$ is integrated from the differential equation (9) with the initial state $\mathbf{P} = \mathbf{0}$. In terms of factor graph, we have derived an IMU factor (i, j):

- **Connected Nodes:** the IMU state $\mathbf{X}_i$ at time-step i and the IMU state the IMU state $\mathbf{X}_j$ at time-step j.
- **Cost function:**
$$r(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{X}_j \ominus F(\mathbf{X}_i, \mathbf{u}_{i:j}) \in \mathbb{R}^{15} \tag{16}$$
- **Covariance matrix:** $\mathbf{P}_{ij}$
- **Measurements:** the pre-integrated matrix $\mathbb{A}$ and the IMU biases $(\hat{\mathbf{b}}_{gi}, \hat{\mathbf{b}}_{ai})$ used in the preintegration.

Then the proposed preintegration elements in (14) results in a closed-form solution of the predicted state $F(\mathbf{X}_i, \mathbf{u}_{i:j})$ and therefore here we provide the closed form of the error function of the proposed IMU factor (16):

$$r(\mathbf{X}_i, \mathbf{X}_j) = \begin{bmatrix} \mathbf{e}_r + J_r^{-1}(\mathbf{e}_r)\mathbf{J}_4(\mathbf{b}_{gi} - \hat{\mathbf{b}}_{gi}) \\ \mathbf{R}_j^\intercal(\mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \mathbf{R}_i\mathbf{J}_2 - \mathbf{v}_j) + \mathbf{J}_5(\mathbf{b}_{gi} - \hat{\mathbf{b}}_{gi}) + \mathbf{J}_4(\mathbf{b}_{ai} - \hat{\mathbf{b}}_{ai}) \\ \mathbf{R}_j^\intercal(\mathbf{p}_i + \mathbf{v}_i\Delta t_{ij} + \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 + \mathbf{R}_i\mathbf{J}_3 - \mathbf{p}_j) + \mathbf{J}_6(\mathbf{b}_{gi} - \hat{\mathbf{b}}_{gi}) + \mathbf{J}_7(\mathbf{b}_{ai} - \hat{\mathbf{b}}_{ai}) \end{bmatrix} \tag{17}$$

where $\mathbf{e}_r = \log(\mathbf{R}_j^\intercal\mathbf{R}_i\mathbf{J}_1) \in \mathbb{R}^3$, $J_r(\cdot)$ and $\log(\cdot)$ are given in Appendix. Note that the proposed error function is linear to all variables except $\mathbf{R}_i$ and $\mathbf{R}_j$. Later we will use this linear property to design the optimization algorithm.

2.2. Vision Factor

The conventional vision factor employs the re-projection error as the cost function, which is

$$\pi(\mathbf{K}\mathbf{R}_c^\intercal(\mathbf{f} - \mathbf{p}_c)) - \mathbf{uv} \tag{18}$$

where $\pi(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection function, $(\mathbf{R}_c, \mathbf{p}_c) = (\mathbf{R}, \mathbf{p})\mathbf{T}_{IC} \in \mathbb{SE}(3)$ is the camera pose, $\mathbf{K}$ is the camera calibration matrix, $\mathbf{T}_{IC} \in \mathbb{SE}(3)$ is the transformation from camera to IMU and $\mathbf{uv} \in \mathbb{R}^2$ is the pixel observation for the map point $\mathbf{f} \in \mathbb{R}^3$. However, the zero re-projection error just implies that the predicted vector is parallel to the measured , which possibly results in a large number of local minimums. To alleviate this shortcoming, we employ the directional error as the cost function, resulting in a different vision factor. We present this improvement with more details in Figure 2.

- **Connected Nodes:** the IMU state $\mathbf{X}_i$ at time-step i and the map point f
- **Cost function:**
$$g(\mathbf{X}, \mathbf{f}) = N(\mathbf{R}_c^\intercal(\mathbf{f} - \mathbf{p}_c)) - d(\mathbf{uv}). \tag{19}$$
where $N(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ for $\mathbf{x} \in \mathbb{R}^3$ and $d(\mathbf{uv}) = N(\mathbf{K}^{-1}\mathbf{UV})$ ($\mathbf{UV} = \begin{bmatrix}\mathbf{uv}, 1\end{bmatrix}$). Note the directional error can be seen as the normalized vector between map point and camera center, which project the map point into a unit sphere. Thus, unlike the projection error in (18) which is an unbounded factor,

the directional error is bounded into the range of $[-2, 2]$, which is friendly to the convergence of the algorithm.

- **Covariance matrix:** $\sigma \mathbf{I}_3$

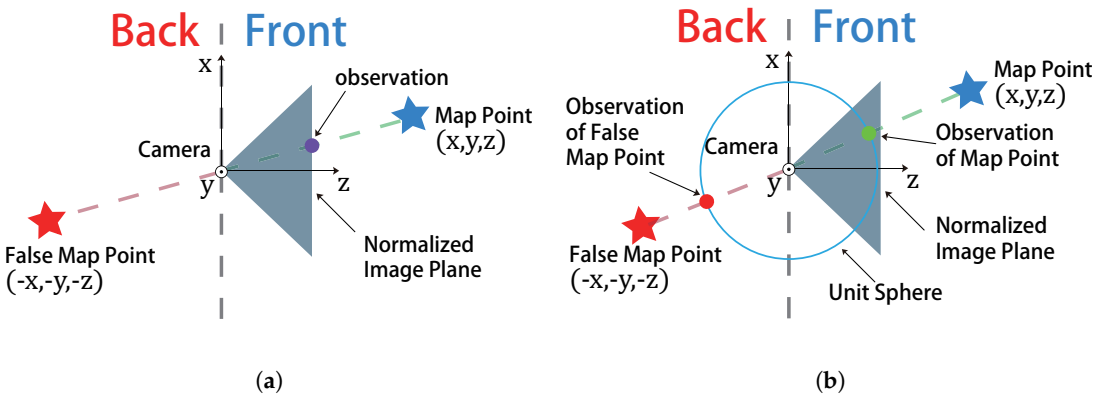


Figure 2. The difference between projection error and directional error. In projection error, since $\pi(x, y, z) = (x/z, y/z)^T$ both the map point (in front of camera) and the false map point (in back of camera) would have the same observation, which can easily leads the algorithm falls into the local minimum. However the directional error employed by our algorithm, which normalized the direction vector between map point and camera center, can have different observations between the map points in the front and back, even their direction vectors is parallel with each other. (**a**) projection error employed by conventional vision factor; (**b**) directional error of our vision factor.

2.3. Nonlinear Least Squares Form

In our proposed system, optimization is used to correct the error due to sensor noise. Given all IMU measurements $\mathbf{u}$ and camera measurements $\mathbf{z}$ along the trajectory, the MAP estimate is

$$\begin{aligned} \mathcal{X}^* &= \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathbf{z}, \mathbf{u}) \\ &= \arg\max_{\mathcal{X}} \prod_k p(\mathbf{X}_k|\mathbf{X}_{k-1}, \mathbf{u}_{k-1:k}) \prod_{k,l} p(\mathbf{X}_k, \mathbf{f}_l|\mathbf{z}_{k,l}) \end{aligned} \tag{20}$$

where $\mathcal{X} = \{\mathbf{X}_k, \mathbf{f}_l\}$ includes the observed map points and the IMU states from time step 0 to N. Note that $p(\mathbf{X}_k|\mathbf{X}_{k-1}, \mathbf{u}_{k-1:k})$ and $p(\mathbf{X}_k, \mathbf{f}_l|\mathbf{z}_{k,l})$ correspond to the IMU factor and the vision factor, as discussed in Section 2.1 and Section 2.2. Based on the theory of factor graph, the optimization problem above can be abstracted into a graph (Figure 3) that consists of nodes ($\mathbf{X}_i$ and $\mathbf{f}_l$) and factors ($\mathbf{r}$ and $\mathbf{g}$). The MAP estimate inference can be transformed into the following nonlinear least squares problem:

$$\begin{aligned} \mathcal{X}^* &= \arg\min_{\mathcal{X}} \sum_i \|r(\mathbf{X}_i, \mathbf{X}_{i+1})\|^2_{\mathbf{P}^{-1}_{i,i+1}} + \sum_{i,l} \|g(\mathbf{X}_i, \mathbf{f}_l)\|^2_{\sigma^{-1}\mathbf{I}_3} \\ &= \arg\min_{\mathcal{X}} \|h(\mathcal{X})\|^2 \end{aligned} \tag{21}$$

Different from the standard least squares problem, the state space of the problem above is non-Euclidean space and thus we integrate the "lift-retraction" strategy into the conventional Gauss-Newton method for solving optimization, which has been summarized in Algorithm 1. Note that $\boxplus$ in Algorithm 1 is *user-defined* and we choose

$$\begin{aligned} \mathcal{X} \boxplus \{(\mathbf{e}_i, \mathbf{e}_l)\} &= \{(\mathbf{X}_i, \mathbf{f}_l)\} \boxplus \{(\mathbf{e}_i, \mathbf{e}_l)\} \\ &= \{(\mathbf{X}_i \oplus \mathbf{e}_i, \mathbf{f}_l + \mathbf{e}_l)\} \end{aligned} \tag{22}$$

where $\oplus$ is given in (11).

Algorithm 1: Solving Equation (21) by Using the Gauss-Newton Algorithm

Input: the initial guess $\mathcal{X}^{(0)}$ and the retraction $\boxplus$.
Output: the local minimum $\mathcal{X}^*$
Process:
$\mathcal{X}^* \leftarrow \mathcal{X}^{(0)}$;
while $\mathcal{X}^*$ *does not converge* **do**
solving the normal equation

$$\mathbf{H}\Delta\mathbf{x}_{gn} = -\mathbf{F}h(\mathcal{X}^*) \tag{23}$$

where $\mathbf{F} := \frac{\partial h(\mathcal{X}^* \boxplus \mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{0}}$ and $\mathbf{H} := \mathbf{F}^\intercal\mathbf{F}$.;
Update: $\mathcal{X}^* \leftarrow \mathcal{X}^* \boxplus \Delta\mathbf{x}_{gn}$;

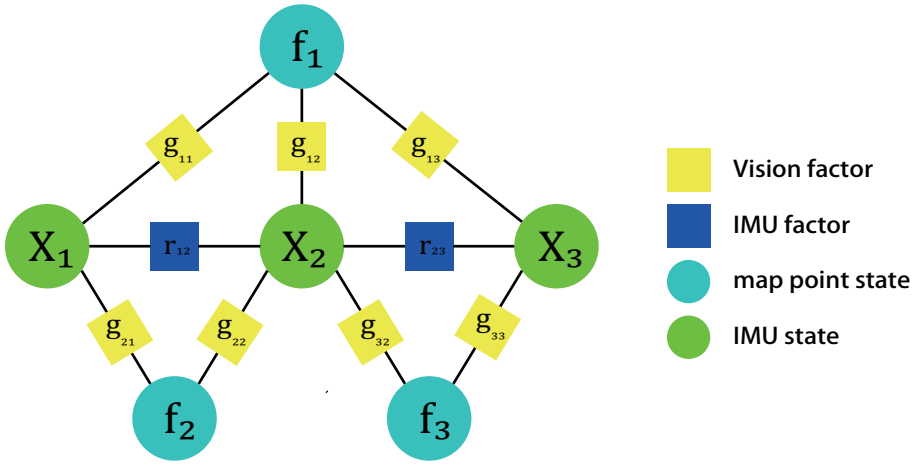


Figure 3. A VINS graph with 3 IMU states and 3 map points. The notation $\mathbf{X}_i$ represents the IMU state at time-step i, $\mathbf{f}_i$ represents the map point i. The notation $\mathbf{g}_{ij}$ represents the vision factor and $\mathbf{r}_{ij}$ stands for the IMU factor. Then the objective cost function for the VINS solution should be adding all factors ogether.

Optimization can provide a relatively accurate estimation for visual construction and the IMU state. However, the Cholesky decomposition used in solving the normal equation (23) suffers from the $O(\det(\bar{\mathbf{H}})^3)$ complexity, where $\bar{\mathbf{H}}$ has the same sparsity of $\mathbf{H}$ and only contains 1 and 0. To alleviate this, we present a novel way to solve (21), which employs the overlooked partial linear structure of (21) and the local observability of VINS. The related details will be given in Section 3.

3. Visual Inertial SLAM Algorithm

Our system is inspired by ORB-SLAM [23] which simultaneously runs the tracking thread and the local mapping thread in real-time.

3.1. Tracking

The tracking thread is in charge of estimating the latest IMU state, which involves twice optimization. In the first optimization, the initial value is given by the IMU preintegration. Then we search for the map points observation by 3D points' projection. Finally, we perform the small-size optimization (24) which is solved by Algorithm 2. The optimization can be seen as an extension of the pose-only bundle adjustment in the ORB-SLAM [23]. Different with [23], the state variables brought by the IMU factor has been considered. We separate the state vector into two groups and optimized

them separately. In the first step, we only update $\mathbf{R}_i$ and $\mathbf{p}_i$ thus we can avoid the large drift caused by the low-cost IMU sensor. Secondly, the optimization turns into a linear least squares problem w.r.t the $(\mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$. We describe this solution with more mathematical detail in the Remark 2.

After the first optimization, we perform a guided search of the map points from last frame. A wider search of the map points will be used if not enough matches are found. For efficient and robust data association, we use the projection method from the frames in the co-visibility graph to the current camera frame to perform feature correspondence. In order to keep the computational complexity bounded, we only deal with the keyframes in the local mapping thread. Therefore, there is a mechanism in the end of the threading thread that decides whether the current frame is a new keyframe. To insert a new keyframe, all the following conditions must be met: (1) More than 5 cm have been passed from the last keyframe; (2) More than 20 frames have been passed from last keyframe; (3) Current frame tracks at least 50 points and the number of common points between current frame and last keyframe should be less than 90% of last keyframe. The last condition ensures a visual change condition, and the 1st and 2nd condition will also reduce the number of unnecessary keyframes. We will also send a waiting signal to stop local mapping thread, so it can process the new keyframe as soon as possible. The framework of tracking is summarized in Figure 4.

Remark 2. *To quickly output the estimate* $\mathbf{x}_i$*, we employ a small-size optimization (24) instead of the full optimization:*

$$\begin{aligned} \mathbf{x}_i^* &= \arg\min_{\mathbf{x}_i} \|h(\mathbf{x}_i)\|^2 \\ &= \arg\min_{\mathbf{x}_i} \|r(\mathbf{X}_{i-1}, \mathbf{X}_i)\|^2_{\mathbf{P}^{-1}_{i-1,i}} + \sum_l \|g(\mathbf{X}_i, \mathbf{f}_l)\|^2_{\sigma^{-1}\mathbf{I}_3} \end{aligned} \tag{24}$$

where $\mathbf{x}_{i-1}$ *is the previous IMU state,* $\mathbf{f}_l$ *denotes the map point observed in the current step. For efficient estimation, both* $\mathbf{x}_{i-1}$ *and* $\mathbf{f}_l$ *are fixed in (24). In addition, an ignorable property of (24) is that given the current pose* $(\mathbf{R}_i, \mathbf{p}_i)$*, the optimization becomes linear least squares problem w.r.t.* $(\mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$*. Therefore, we employ the* ***separability*** *trick for solving (24), which is summarized in Algorithm 2.*

Algorithm 2: Optimization (24) in Tracking

Input: the initial guess $\mathbf{x}_i = (\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$.
Output: the local minimum $\mathbf{x}_i^*$
Process:
$\mathbf{x}_i^* \leftarrow \mathbf{x}_i$;
while $\mathbf{x}_i^*$ *does not converge* **do**
  Extract $(\Delta\mathbf{R}, \Delta\mathbf{p}) \in \mathbb{R}^6$ from the normal equation

$$\mathbf{H}\Delta\mathbf{x}_{gn} = -\mathbf{F}h(\mathbf{x}_i^*) \tag{25}$$

  where $\Delta\mathbf{x}_{gn} = (\Delta\mathbf{R}, \Delta\mathbf{p}, \Delta\mathbf{v}, \Delta\mathbf{b}_g, \Delta\mathbf{b}_a)$, $\mathbf{F} := \frac{\partial h(\mathbf{x}_i^* \boxplus \mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{0}}$ and $\mathbf{H} := \mathbf{F}^\intercal\mathbf{F}$.;
  Update pose: $(\mathbf{R}_i^*, \mathbf{p}_i^*) \leftarrow (\mathbf{R}_i^*, \mathbf{p}_i^*) \oplus (\Delta\mathbf{R}, \Delta\mathbf{p})$;
  Update $(\mathbf{v}_i^*, \mathbf{b}_{gi}^*, \mathbf{b}_{ai}^*)$ from the linear least squares (24) in which $(\mathbf{R}_i^*, \mathbf{p}_i^*)$ is fixed;

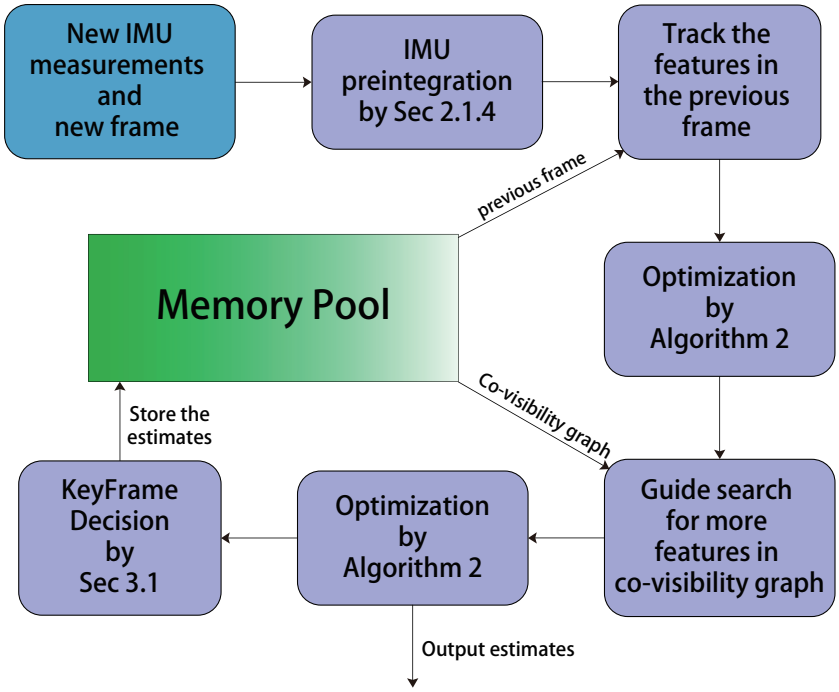


Figure 4. The framework of the Tracking thread.

3.2. Local Mapping

Once a keyframe is inserted from the tracking thread, the local mapping thread will begin its work that includes creating map points, deleting map points, deleting keyframes and performing optimization. The flowchart of the local mapping thread is presented in Figure 5, and we also add a graph to present the local mapping thread in Figure 6.

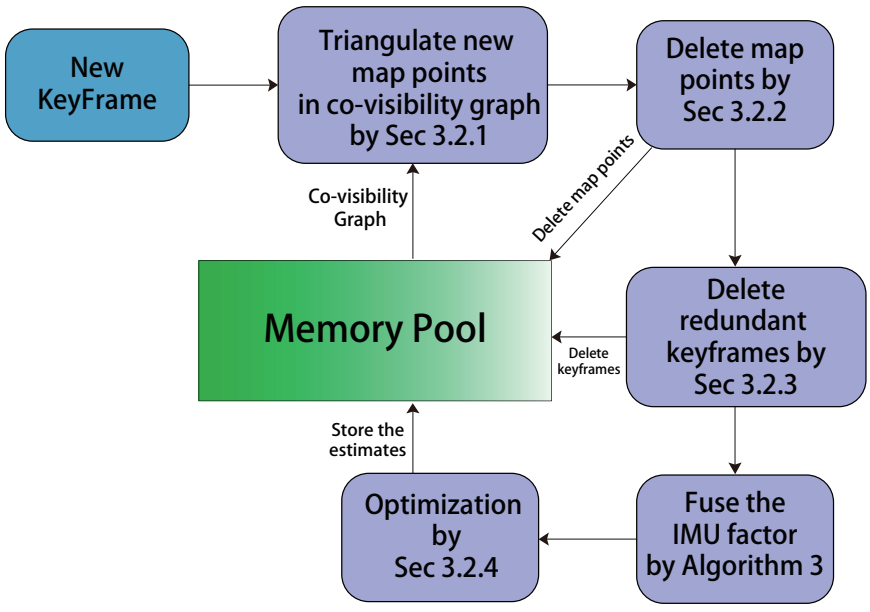


Figure 5. The framework of the Local Mapping thread.

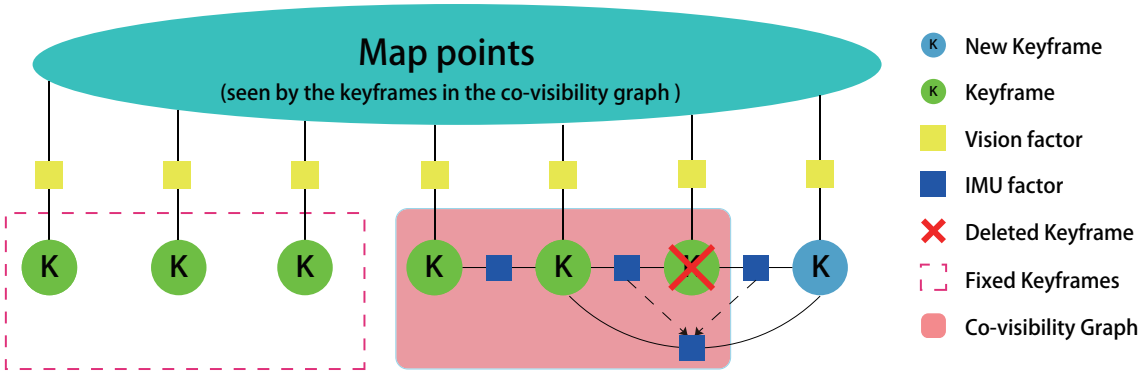


Figure 6. Graph illustration with 7 keyframes for the local mapping thread. Keyframes inside the co-visibility graph (red transparent area) will be connected by IMU factors, and their IMU state will be optimized by the local mapping thread. Other keyframes in the memory pool which also observe the map points will remain fixed (red dot rectangle). The graph also illustrates the IMU factor's evolution when we delete a keyframe. Two IMU factors which are connected to the deleted keyframe will be fused into a new IMU factor by the Algorithm 3.

3.2.1. Creat Map Points

When the local mapping thread gets a new keyframe, new map points observed in the new keyframe and the local keyframes will be created by triangulation. The following are the main steps. First, the projection method from the local keyframes is used to search the feature correspondences. The search is performed according to the time order, and it begins from last keyframe and stops once it fails to get a match. With the new feature correspondences from the search, we then calculate the coordinates of new map points by using the fast linear triangulation. In order to get rid of spurious data association, we only keep the new map points that are observed at least three times. Finally, the co-visibility graph will be updated by adding the undirected edges between the keyframes that share the same map points.

3.2.2. Delete Map Points

Considering that outliers or incorrect feature correspondences will significantly affect the system performance, map points culling is needed before optimization. In the local mapping thread, we check the epipolar constraint and reprojection error of each map point for each keyframe which observes this point. In addition, we also check the parallax angle of each point. If the maximum parallax value is below a threshold, the map point will be removed. In this step, only the map points in the local map are processed.

3.2.3. Delete KeyFrames

Deleting the redundant keyframes is beneficial for optimization, which saves the computational time. We discard keyframes whose 90% of map points have been seen in at least other three keyframes. When deleting a keyframe, we need to integrate two IMU factors (connected to this keyframe) into one IMU factor (connected to the last keyframe and the next keyframe). According to the theory of linear system, we derived an integration algorithm, summarized in Algorithm 3. Figure 6 shows the keyframe process in optimization graph, from which we can easily see the way of IMU factor fusion when delete a keyframe.

Algorithm 3: The Fusion of Two Consequential IMU Factors

Input: two consequential IMU factors (i, j) and (j, k)
Output: IMU factor (i, k)
Process:
Connected Nodes: the IMU state $\mathbf{X}_i$ and the IMU state the IMU state $\mathbf{X}_k$.
Cost function:

$$r(\mathbf{X}_i, \mathbf{X}_k) = \mathbf{X}_k \ominus F(\mathbf{X}_i, \mathbf{u}_{i:k}) \in \mathbb{R}^{15} \tag{26}$$

Covariance matrix: $\mathbf{P}_{ik} = \mathbb{A}_{j,k}\mathbf{P}_{ij}\mathbb{A}_{j,k}^{\mathsf{T}} + \mathbf{P}_{jk}$.
Measurements: the pre-integrated matrix $\mathbb{A}_{ik} = \mathbb{A}_{jk}\mathbb{A}_{ij}$ and the IMU biases $(\hat{\mathbf{b}}_{gi}, \hat{\mathbf{b}}_{ai})$.

3.2.4. Optimization

The last step of the local mapping thread is the optimization (21) with the nodes:

(a) the latest IMU state $\mathbf{x}_i$ and all IMU states $\mathbf{x}_j$ in the co-visibility graph (w.r.t. $\mathbf{x}_i$);
(b) all map points $\mathbf{f}_l$ observed by $\mathbf{x}_j$ in the co-visibility graph (w.r.t. $\mathbf{x}_i$);
(c) all IMU state $\mathbf{x}_k$ that observes the map points in (b). Note that these variables are fixed in the optimization.

The involved factors are:

- The IMU factors that connects the consecutive IMU states in (a).
- The vision factors that connects the IMU states in (a) or (b) and the map points in (c).

To maintain a consistent estimate, we fix the IMU states in (b). Typically, the naive implementation of the optimization here suffers from the $O((15n)^3)$ computational complexity in solving the *reduced* normal equation, where n is the number of IMU states in (a). Similar to the *separability* trick in Algorithm 2, we also use *separability* strategy on the optimization here so that the computational complexity can be reduced to $O((6n)^3)$, which is given in the following Algorithm 4.

Algorithm 4: Optimization in Local Mapping

Input: the initial guess $\mathcal{X}$ that consists of $\mathbf{x}_i = (\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$ in (a) and $\mathbf{f}_l$ in (b)
Output: the local minimum $\mathcal{X}^*$
Process:
$\mathcal{X}^* \leftarrow \mathcal{X}$;
while $\mathcal{X}$ *does not converge* **do**
  Fix all $\mathbf{v}_i^*, \mathbf{b}_{gi}^*, \mathbf{b}_{ai}^*$, employ the *Schur* trick and extract all $(\Delta\mathbf{R}_i, \Delta\mathbf{p}_i) \in \mathbb{R}^6$ from the normal equation

$$\mathbf{H}\Delta\mathbf{x}_{gn} = -\mathbf{F}h(\mathcal{X}^*) \tag{27}$$

  Update pose: $(\mathbf{R}_i^*, \mathbf{p}_i^*) \leftarrow (\mathbf{R}_i^*, \mathbf{p}_i^*) \oplus (\Delta\mathbf{R}_i, \Delta\mathbf{p}_i)$;
  Update map point $\mathbf{f}_l$ via the back-substitution with $\{(\Delta\mathbf{R}_i, \Delta\mathbf{p}_i)\}$.
  Update $(\mathbf{v}_i^*, \mathbf{b}_{gi}^*, \mathbf{b}_{ai}^*)$ from the linear least squares (21) in which all pose and map point is fixed;

4. Initialization

In this section, we propose a novel initialization method that provides a robust estimate at the beginning stage. The initialization is significant to the visual-inertial SLAM system due to the nonlinearity in optimization. Inspired by the linear property of the variables $(\mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$ in the IMU factor, we propose a linear least square that can estimate the scale, the velocity, the IMU biases and their covariance matrix. To achieve the reliable estimates and handle the case of poor observability,

the linear estimator will keep running until the uncertainty is lower than a threshold. The whole initialization scheme is presented by Figure 7.

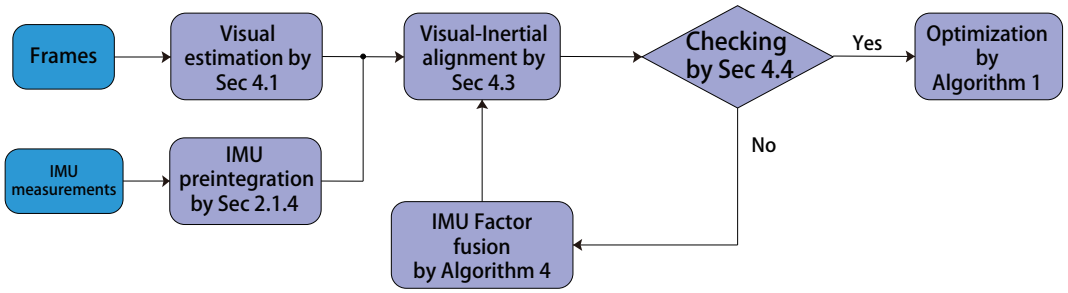


Figure 7. The framework of the initialization.

4.1. Visual Estimation

At the first step, we employ the pure monocular ORB-SLAM [23] to produce the estimates of the frame and their IMU body poses. Note that the absolute scale s is unobservable in the pure visual odometry. The output $(\mathbf{R}_i, \mathbf{P}_i) \in \mathbb{SE}(3)$ from the pure visual odometry is up to the scale s, i.e.,

$$(\mathbf{R}_i, \mathbf{P}_i) = (\mathbf{R}_i, \frac{\mathbf{p}_i}{s}) \tag{28}$$

for $i = 0, 1, \cdots, N$, where $s \in \mathbb{R}$ is the undetermined scale.

4.2. IMU Preintegration

Along with the visual estimation, we also perform the IMU preintegration as shown in Section 2.1. This step will output N IMU factors: the IMU factors $(0, 1)$, $(1, 2)$, $\cdots$, $(N-1, N)$. Note that here the nominal IMU biases used for the preintegration of (12) and (9) are zeros.

4.3. Visual-Inertial Alignment

After visual estimation and IMU preintegration, we perform the visual-inertial alignment to roughly estimate the scale, gravity, velocity, IMU biases. First of all, we substitute (28) into the IMU cost function $r(\mathbf{X}_{i-1}, \mathbf{X}_i)$ and then we can see that the variables s, $\mathbf{g}$, $(\mathbf{v}_i, \mathbf{b}_{gi}, \mathbf{b}_{ai})$ and $(\mathbf{v}_{i-1}, \mathbf{b}_{g,i-1}, \mathbf{b}_{a,i-1})$ are linear in this the term $r(\mathbf{X}_{i-1}, \mathbf{X}_i)$. Fixing the variables $(\mathbf{R}_i, \mathbf{P}_i)$ for $i = 0, 1, \cdots, N$, the MAP problem from (21) becomes

$$\begin{aligned} X^* &= \arg\min_X \sum_i \|r(\mathbf{X}_i, \mathbf{X}_{i+1})\|^2_{\mathbf{P}^{-1}_{i,i+1}} \\ &= \arg\min_X \|\bar{h}(X)\|^2 \end{aligned} \tag{29}$$

where $X = (s, \mathbf{g}, \mathbf{v}_0, \mathbf{b}_{g0}, \mathbf{b}_{a0}, \cdots, \mathbf{v}_N, \mathbf{b}_{gN}, \mathbf{b}_{aN})$. Note that here $\bar{h}(X)$ is almost linear to the variable block X. Thus we can straightforwardly obtain the solution X^* of (29) using the linear least square. However, the linear solution for (29) does not consider the magnitude $\|\mathbf{g}\| = 9.8$ and thus it easily gets ill-conditioned. If we consider the magnitude constraint of $\mathbf{g}$, the linear optimization (29) becomes

$$\begin{aligned} &\min_X \|\bar{h}(X)\|^2 \\ &st: \mathbf{g}^\intercal \mathbf{g} = 9.8^2 \end{aligned} \tag{30}$$

The new optimization problem (30) is quadratically constrained quadratic program (QCQP) problem, which is a convex problem. It is well known that the local minimum in convex optimization

is always a global minimum. Thus we convert (30) to a equivalent unconstrained form formulated in factor graph

$$\min_{X} \|\bar{h}(X)\|^2 \tag{31}$$

with a corresponding retraction

$$X \oplus \mathbf{e} = (s + e_s, \exp(\mathbf{C}\mathbf{e}_g)\mathbf{g}, \mathbf{v}_0 + \mathbf{e}_{v0}, \mathbf{b}_{g0} + \mathbf{e}_{bg0}, \mathbf{b}_{ba0} + \mathbf{e}_{ba0}, \cdots, \mathbf{b}_{aN} + \mathbf{e}_{baN}) \tag{32}$$

where $\mathbf{e} = \left[e_s, \mathbf{e}_g, \mathbf{e}_{v0}, \mathbf{e}_{bg0}, \mathbf{e}_{ba0}, \cdots, \mathbf{e}_{vN}, \mathbf{e}_{bgN}, \mathbf{e}_{baN}\right] \in \mathbb{R}^{9N+12}$, $\mathbf{e}_g \in \mathbb{R}^2$ and $\mathbf{C} \in \mathbb{R}^{3\times 2}$ can be regarded as the null space of $\mathbf{g}$. The use of (32) can grantee that the magnitude of $\mathbf{g}$ keeps unchanged after optimization.

4.4. Checking

It is well-known that a good visual-inertial alignment requires sufficiently motion. For robust estimates, we expect a smart checking step that is in charge of deciding if the estimate X^* from last step (Section 4.3) is safe or not. Here we adopt a value to quantify the accuracy/uncertainty of the estimate X^*, which is the worst-case estimation error [41,42]

$$\lambda_{max}(\bar{\mathbf{H}}^{-1}) \tag{33}$$

where $\bar{\mathbf{H}}$ is the information matrix of scale and gravity, extracted from the Hessian matrix for the optimization problem (31), evaluated at the point X^*. Note that the larger value of $\lambda_{max}(\bar{\mathbf{H}}^{-1})$, larger uncertainty of gravity and scale.

4.5. Optimization

If $\lambda_{max}(\bar{\mathbf{H}}^{-1})$ is more than a threshold σ_{int}, the system will accept the estimate X^*. To refine the estimate X^*, we perform the optimization process (21) with all IMU preintegration and visual measurements. After this step, we have finished the whole initialization.

4.6. IMU Factor Fusion

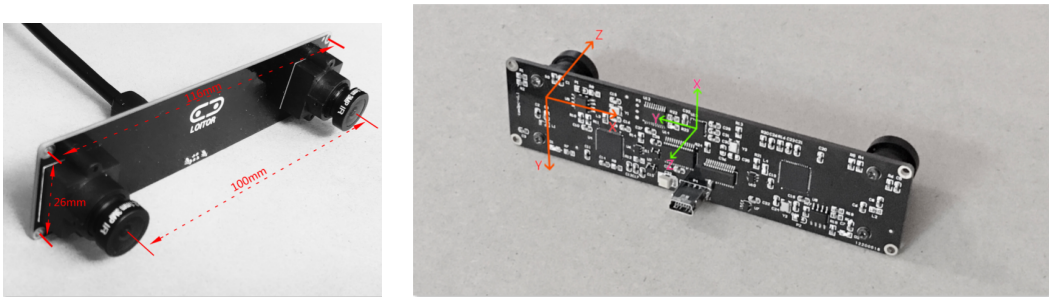
If $\lambda_{max}(\bar{\mathbf{H}}^{-1})$ is less than the threshold σ_{int}, the system will reject the estimate X^* and wait a time-step for reinitialization. The reinitialization will be boosted with all measurements from time-step 0 to time-step $N+1$. Before the reinitialization, we perform IMU fusion of the IMU factors $(N-2, N-1)$ and $(N-1, N)$ to bound the size of the IMU factors. Note that the fusion algorithm is given in Algorithm 3.

5. Implementation Details and Results

The algorithm is implemented via C++11 code with ceres-solver [25] for nonlinear optimization framework. The proposed method runs in real-time (20 Hz) for all experiments on a standard computer (Intel Pentium CPU G840, 2.8 GHz, Dual-Core, 8 GB RAM). We test and evaluate our monocular visual-inertial SLAM system in both the low-cost, off-the-shelf visual-inertial sensor (Figure 8) and the EuRoC dataset [43].

At the beginning of the tracking thread (in Section 3.1), we select keypoints that are well-distributed in the current image. First we detect FAST corners in the 4 pyramid levels of the image. We then split the image into the 32×32 blocks. For each block, we calculate the average Shi-Tomasi score [44] for the FAST corners inside this block. Then we filter out those FAST corners below a specific threshold and calculate the number of the rest FAST corners in each block. If there is a block in which the number of FAST corners is very small (below the 20 percent of the median value of all blocks), we set the threshold to be half of the original one to get more FAST corners. If there is a block that does not contain any FAST corner, we split the image into the 16×16 blocks and repeat the selection

steps above. After extracting the FAST corners, we then calculate the orientation and ORB descriptor for each retained corner. This stage takes about 17 ms on our computer.



(**a**) (**b**)

Figure 8. Loitor Sensor and coordinate system of IMU and left camera. Note we only use the left camera and the IMU sensor for testing our monocular visual inertial SLAM. For more details about the Loitor Sensor, see Loitor's SDK page: https://github.com/loitor-vis. (**a**) Loitor Sensor; (**b**) The coordinate systems of IMU sensor and left eye camera.

After obtaining these keypoints with descriptors, we use the preintegrated IMU measurements (Section 2.1) to get the initial guess of the IMU state (1). In order to deal with the extreme case for the low-cost accelerometer, we filter out those accelerometer readings that are more than 50 times of the last reading. Then we start to perform the guided search of map points in the tracking thread (Section 3.1). Note that the feature correspondences in this step is coupled with the invariance property of the ORB descriptor such that the keypoints in the current frame can be matched with some earlier observations. In addition, we use the efficient subspace dog-leg algorithm in ceres-solver [25] to implement the nonlinear optimization (24). Multi-threads to compute the cost functions and jacobians are used to speed up the system.

We pay more attention about the outliers in the local mapping thread (Section 3.2). In order to gain robust performance, huber loss function with the scale value of 0.2 is used in the vision factors. To get rid of the effects caused by outliers, we first optimize with the huber loss function and then delete the vision factors whose cost function value is more than 0.2. We also check the estimated depth between each map points and keyframes. Map points with negative depth value will be seen as outliers and deleted. After deleting those map points that are outliers, we perform the nonlinear optimization without huber loss function.

5.1. Initilization Implementation

The proposed VINS initialization is evaluated in the in the sequence *V*1_01_*easy*. Because the robust estimates from visual odometry also need enough information, we perform the $\mathbb{SE}(3)$ estimates of the IMU poses at the first 3 s with the visual initialization from ORB-SLAM [23] and then implement the initialization method presented in last section. Figure 9 shows the uncertainties of gravity and scale, which converges after 11 s. The convergence means that the information is enough and then the system can work with a reliable initial estimate. The novelty of our method is

- Our method jointly optimizes the scale, gravitational vector, IMU biases, IMU velocity with proper covariance matrix from preintegration.
- Our method subtly uses the knowledge of the magnitude of the gravitational vector such that the ambiguity between the gravitational vector and the accelerometer bias can be avoided.
- We have a criterion to check whether the estimates for initialization is robust or not.

Since the proposed initialization method is convex which means a unique minimum solution, we optimize the intialization with Gauss Newton method for faster convergence. The Gauss Newton

method is implemented by our own source code with Eigen C++ library [45]. The time cost for this optimization in initialization method is 23 ms on average. Note here we do not use huber loss function cause there is no outlier in IMU measurements. Neither ransc nor multi-thread implementation is needed. After this initialization module, we scale the poses of cameras and the positions of the map points.

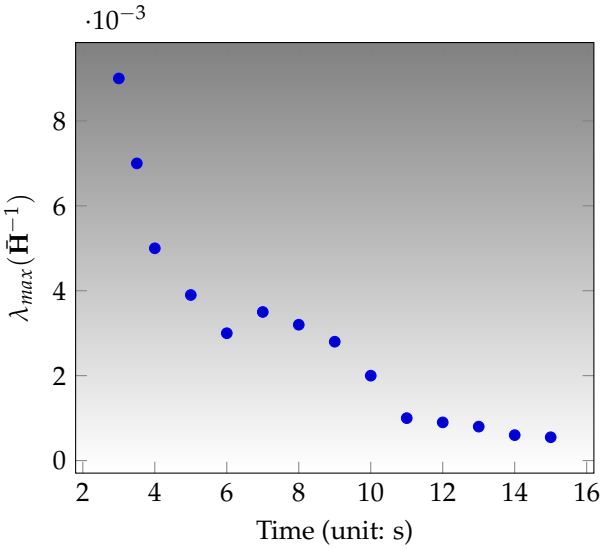
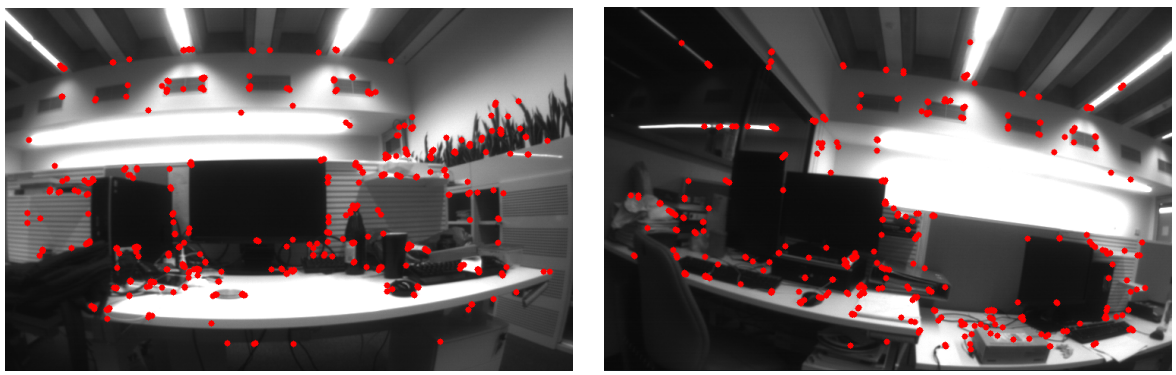


Figure 9. IMU initialization in *V*1_01_*easy*: the uncertainty of gravity and scale.

5.2. Preliminary Test on Low-Cost Hardware

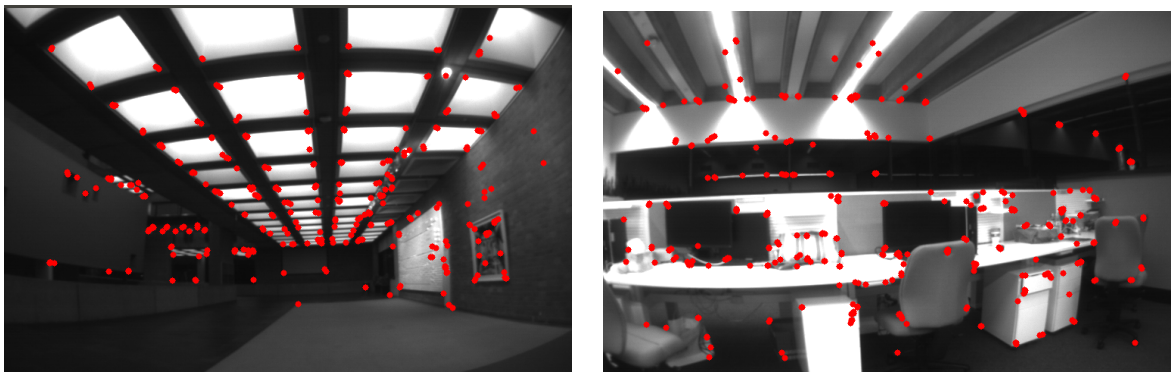
In this subsection, the adopted visual-inertial sensor is the Loitor inertial Stereo camera which is a low-cost device. The Loitor device contains a synchronized global shutter stereo camera which is able to output the 640 × 480 images at the frequency 30 Hz. The device also includes a MPU-6050 IMU with the frequency 200 Hz. The stereo camera and the IMU sensor have been synchronized. This sensor is calibrated by the calibration toolbox Kalibr [46]. Note here although the device contains a stereo camera, we just use the output of the left camera for testing our system. The entire algorithm is implemented in C++ using ROS for acquiring device data.

The algorithm is tested under an indoor scene with random texture. Chess board or any special visual tag is unavailable. The rate of the algorithm is 20 Hz on our computer, with a hand-held Loitor device. Figure 10 shows the well-distributed keypoints in the images. The top view of the estimated trajectory from the proposed system is plotted in Figure 11.



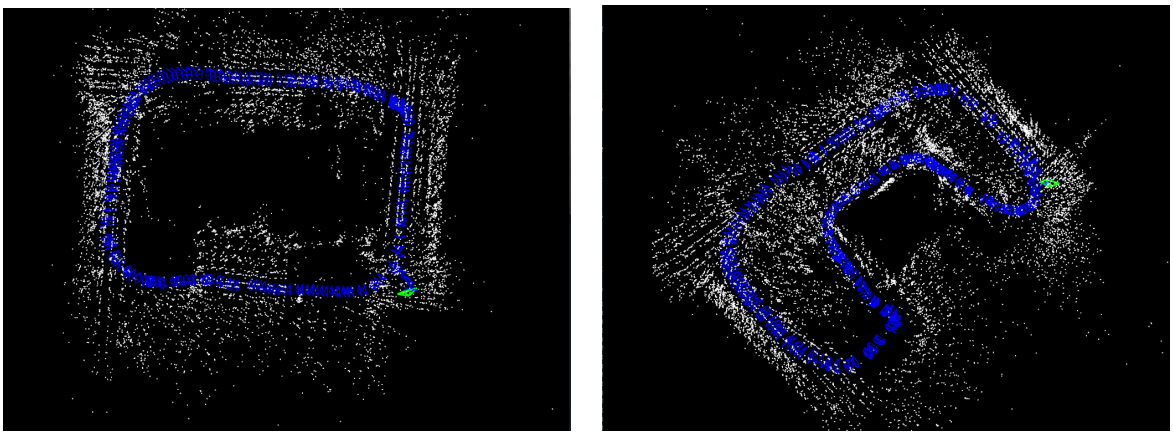
(**a**) (**b**)

Figure 10. *Cont.*



(**c**) (**d**)

Figure 10. The distribution of keypoints in the images. (**a**,**b**,**d**) are taken in computer rooms and (**c**) is taken in the meeting hall.



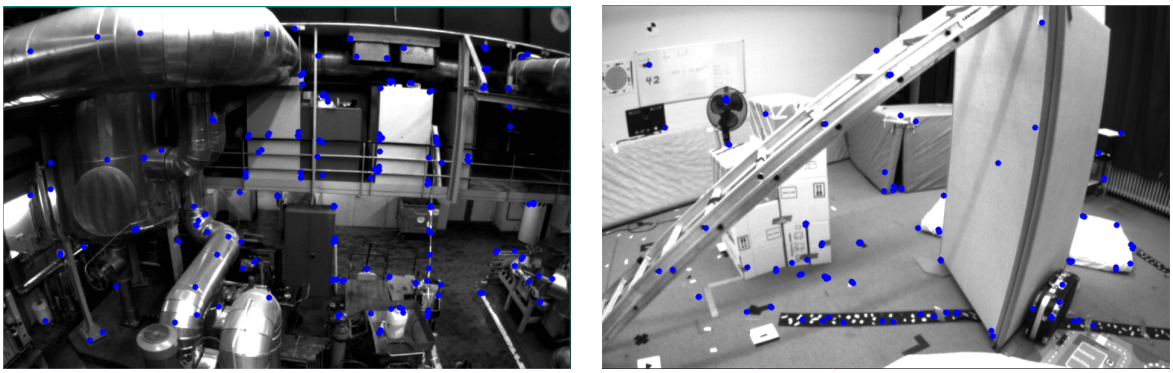
(**a**) (**b**)

Figure 11. The top view of two estimated trajectories (in blue triangle) and map points (in white points) of our system. We can see the trajectory in (**a**) has some drift at the green point. Drift also exists in the image (**b**), but is not obvious. Both the experiments are implemented under an indoor environment with the size of 60 m $\times$ 60 m.

5.3. Evaluation on EuRoC

The accuracy of our Visual-Inertial SLAM is evaluated in the 11 sequences of the EuRoC dataset. The EuRoc dataset provides synchronized global shutter WVGA stereo images at 20 Hz with MEMS IMU measurements at 200 Hz and trajectory ground-truth under different rooms in Figure 12. The dataset was collected by a MAV and ground truth is gained by a Vicon motion capture system which provided 6 DOF(degree of freedom) pose measurements at 100 Hz of a coordinate frame. For more detail we refer to paper [43].

IMU initialization is performed inside the SLAM system. Our system fails to run the sequence $V1\_03\_difficult$ since the visual only SLAM failed to initialize under the extreme movement. For other data sequence, our SLAM algorithm can run in real-time without tracking lost. Table 1 presents the results of RMSE and standard deviation (in terms of translation) for different data sequences in EuRoC dataset. We evaluate the trajectories through the ATE method [47] which align the trajectories first, and then group them by the distance, finally compute the RMSE for each group. We present more details for the comparisons in Figure 13. In Figure 14 we plots some trajectories for our SLAM estimations and the ground truth (in top view).



(**a**) (**b**)

Figure 12. The different rooms in EuRoC data sets. Data sequences of Machine Hall in (**a**) have rich texture while Vicon Room of (**b**) have a lot of white walls which make them difficult for feature tracking.

Table 1. RMSE and Std deviation results for data sequence.

Sequence	RMSE (Unit: m)	Std (Unit: m)
*V*1_01_*easy*	0.0542	0.0194
*V*1_02_*medium*	0.0607	0.0246
*V*1_03_*difficult*	X	X
*V*2_01_*easy*	0.0424	0.0145
*V*2_02_*medium*	0.0430	0.0150
*MH*_01_*easy*	0.1010	0.0459
*MH*_02_*medium*	0.0643	0.0294
*MH*_03_*medium*	0.0632	0.0257
*MH*_04_*difficult*	0.0921	0.0384
*MH*_05_*difficult*	0.1378	0.0348

We compare our proposed system with two state-of-art reasearch: stereo-inertial odometry OKVIS [11] and the VINS-MONO [30] without loop-closure and VINS-MONO with loop closure for completeness. Figure 13 shows the results of three system in terms of RMSE. From the error bar results in (b), (d), (e) and (f) in Figure 13, we can see our algorithm significantly outperforms the state-of-art algorithm VINS-MONO [30] and OKVIS [11] with stereo camera, which can be explained by the following:

- Our proposed IMU factor is more linear and it does not need reintegration when optimization. The cost function of our proposed IMU factor is more linear in terms of the defined retraction $\oplus$. The propagated covariance can better reflect the uncertainty of the physical system.
- The use of the separability trick and the novel vision factor makes convergence faster than the conventional method such that local or global minimum can be reached after few iterations in optimization.
- The use of co-visibility graph in our system can provide edges from current IMU state to the map points observed by the earlier IMU states, Since the data sequences in EuRoC is taken in a single small room, the drone can get the earlier observations easily by turning around, which makes the algorithm with co-visibility graph performs with much better precision.
- The fusion of IMU factors also provides the constraints between two consequential IMU states.

We would let readers know that although our algorithm performs with high precision, it fails to run the V103 data sequence. This data sequence has extreme movement at the beginning which is the main reason for the initialization failure in our system. In comparison, OKVIS [11] with a stereo camera can run without performing initialization which make the algorithm handling this data sequence easily.

However, OKVIS fails to process the V203 data sequences. On the other hand, VINS-MONO [30], use sparse optical flow tracking as an independent front-end module to retrieve data association. Optical flow is a robust way for tracking features in video, which makes the initialization successfully and let the algorithm can process all the data sequences in EuRoC dataset. However, the algorithm suffers from low precision for ignoring the early observations. Besides, optical flow is not accurate for feature tracking in sub-pixel accuracy.

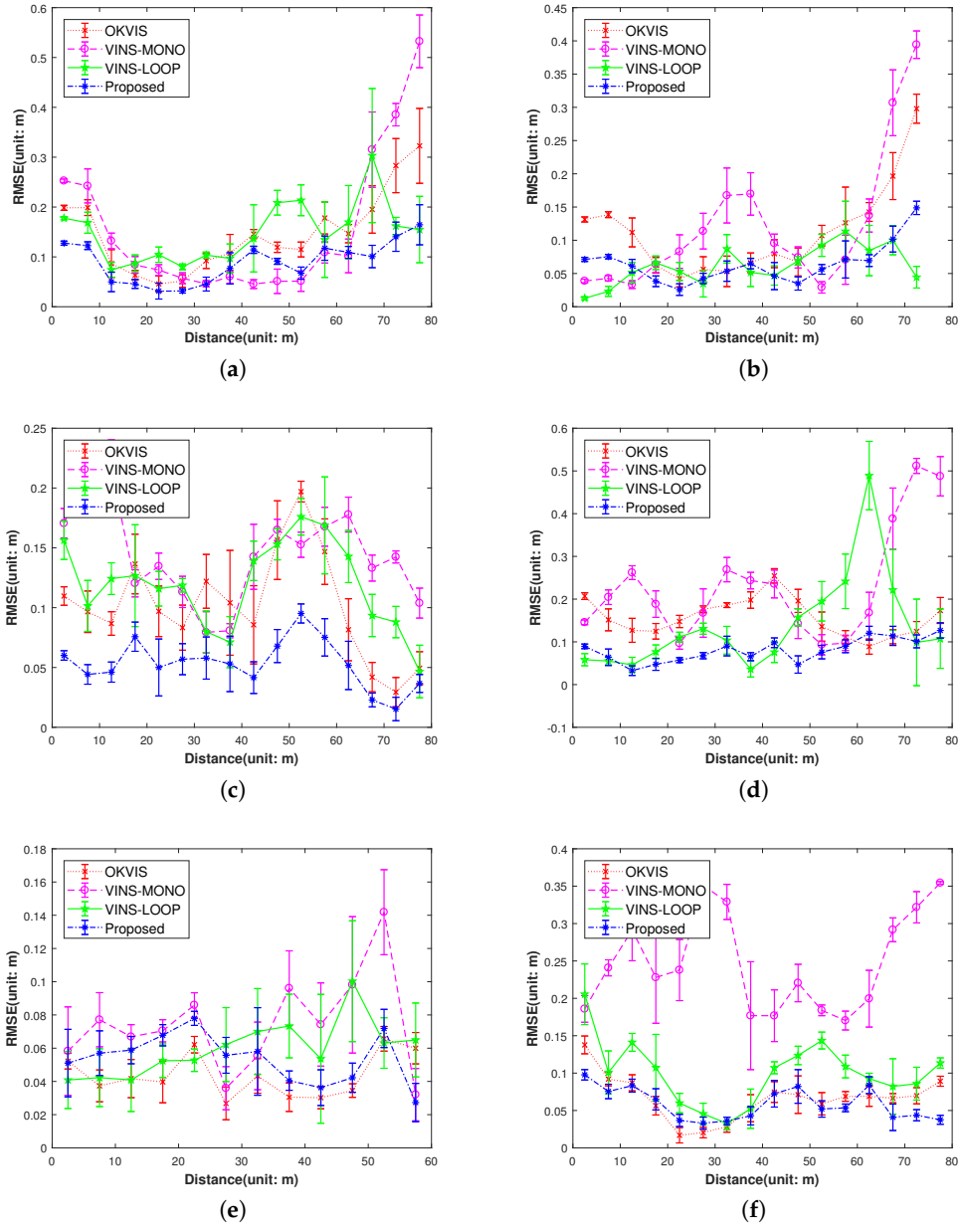


Figure 13. *Cont.*

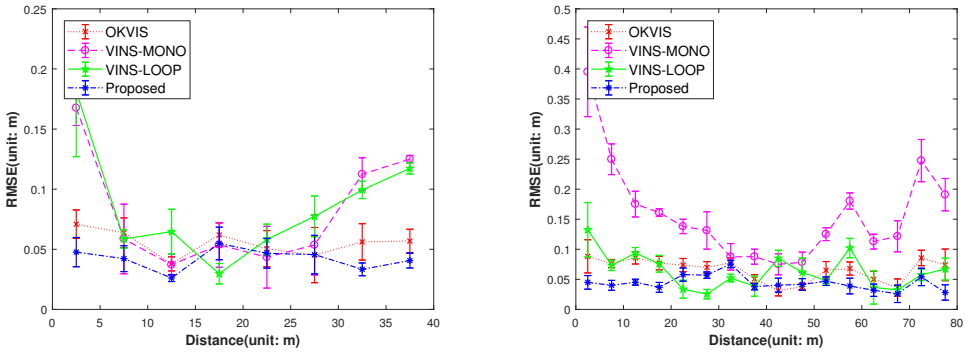


Figure 13. Comparison of the proposed method versus the OKVIS, VINS-MONO and VINS with loop closure (VINS-LOOP). The OKVIS uses a stereo visual-inertial sensor and the VINS-MONO (VINS-LOOP) uses a monocular visual-inertial sensor. Our algorithm has substantial improvement over other two methods in the Machine Hall (MH01-MH04) data sequences, also has comparable result with OKVIS in the rest of data sequences. Note we haven't show the results of V103 and V203, since our algorithm fails to run the V103 and OKVIS fails to run the V203 data sequence. (**a**) *MH*_01_*easy*; (**b**) *MH*_02_*medium*; (**c**) *MH*_03_*medium*; (**d**) *MH*_04_*difficult*; (**e**) *V*1_01_*easy*; (**f**) *V*1_02_*medium*; (**g**) *V*2_01_*easy*; (**h**) *V*2_02_*medium*.

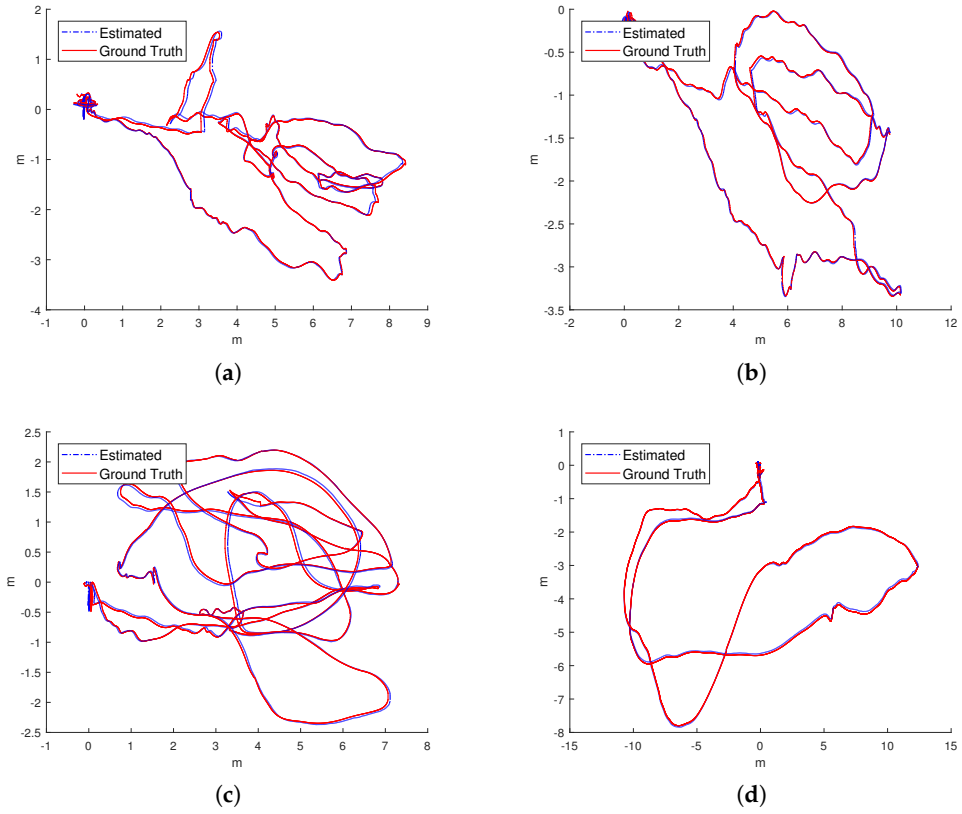


Figure 14. *Cont*.

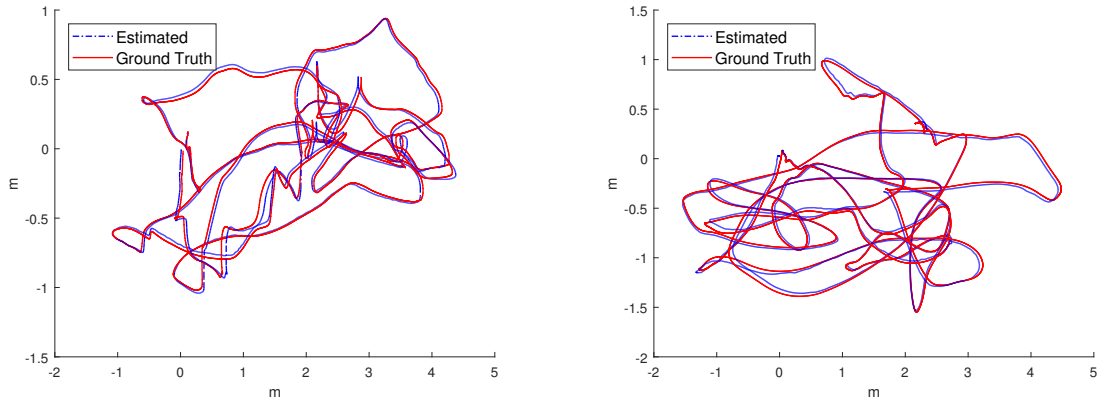


Figure 14. The top-views of estimated trajectory from our proposed approach (blue line) and ground truth (red line) of the dataset. (**a**) *MH_01_easy*; (**b**) *MH_02_medium*; (**c**) *MH_03_medium*; (**d**) *MH_04_difficult*; (**e**) *V1_01_easy*; (**f**) *V1_02_medium*.

6. Discussion and Future Work

In this paper, based on the pure monocular vision ORB-SLAM [23], we present a monocular visual-inertial SLAM system with our new IMU factor, vision factor, initialization. The proposed visual-inertial slam has high precision over the EuRoC dataset. One of the main reason behind this, is the co-visibility graph we employed from the ORB-SLAM [23], since even we found even the stereo ORB-SLAM [23] without IMU can have higher precision than OKVIS [11] with stereo-inertial sensor. The co-visibility graph makes it possible to utilize the early observations while the sliding window based methods ignore them. Our algorithm has substantial improvement on Machine Hall data sequences in EuRoC, since the visual texture is very friendly for ORB feature tracking and co-visibility graph construction. Meanwhile, our algorithm achieves the comparable performance with the state-of-art method OKVIS which use a stereo camera and IMU sensor on the Vicon Room data sequences. In these data sequences we found our algorithm happened to track lost for a few times since they contain a lot of white walls and gray boards which is hard to detect any features on them. We can easily see this effect in Figure 12.

Since the algorithm still use the same front-end module, same keyframe decision with visual ORB-SLAM [23], we still think there is lots of things to do for further improvement. (1) For the front-end, we think direct method, which directly use the gray scale value into optimization, is a promising way since it can use weak feature that just have gradient value. Like in the DSO algorithm [17], by understanding more exposure adjustment in optical camera, the algorithm have surprising precision with impressive robustness. (2) For the vision factor in back-end, we found that our implementation of directional error have higher precision, but it is a bit slower than original bundle adjustment. There will be more comparison with more details between direction and projection vision factor in our future research. Also, we think getting rid of the map points that their parallax are below certain threshold is not reasonable for it lost the rotation information given by those map points. However, map points with low parallax will turns the system into ill-posed since their location is not observable. Therefore, developing a vision factor that can make use of low parallax is essential. (3) Some basic technique like on-line calibration for the sensor, loop closure can also be added into system. Machine learning methods that detect movable objects in visual observation can also be tried.

7. Conclusions

This paper demonstrates a new method for the monocular vision and inertial state estimation algorithm with a real-time implementation. The proposed IMU preintegration not only reaches the state of art efficiency, but also have better linear form which can better capture the correlation of state uncertanties. To increase the speed of the algorithm, the separability trick and the novel vision factor for fast computation was used in both the tracking thread and the local-mapping thread. Thanks to the proposed IMU preintegration with better linearity, the proper weight and the reasonable criterion to check the reliability of the estimates, our initialization method is fast and reliable, which solves a convex optimization with less uncertainty. So far we have build a tightly coupled visual-inertial SLAM system that can run with real-time performance in unknown environment. The future work will be mainly on providing more data to give more insights about the performance of our initialization and seek a better model for the tightly-coupled visual-inertial SLAM's back-end.

Acknowledgments: This project is supported by the National Science Fund of China under Grants 6171136, the Research Fund for the Doctoral Program of Higher Education of China under Grants 20110142110069 and CALT Aerospace Fund. I would also like to give my special thanks to Dr. Teng Zhang from Centre for Autonomous Systems, University Technology of Syndey, who gave me the special insight of IMU preintegration and the evaluation of initialization's robustness.

Author Contributions: Yi liu conceived, designed the experiments and wrote the paper; Zhong Chen and Jianguo Liu contributed analysis tools; Hao Wang and Wenjuan Zhen analyzed the data. Yi liu performed the experiments.

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

Appendix A. Math

Here we provide the mathematical functions used in this paper. For more details, see [48]. The operator $S(\cdot)$ transforms a 3-dimensional vector to a 3×3 matrix:

$$S(\mathbf{x}) = \begin{bmatrix} 0 & -x_2 & x_3 \\ x_2 & 0 & -x_1 \\ -x_3 & x_1 & 0 \end{bmatrix} \tag{A1}$$

for $\mathbf{x} = \begin{bmatrix} x_1, x_2, x_3 \end{bmatrix}^\intercal$. The exponential mapping $\exp(\cdot)$ transforms a 3-dimensional vector to a 3×3 rotation matrix:

$$\exp(\mathbf{x}) = \mathbf{I}_3 + \frac{\sin(\|\mathbf{x}\|)}{\|\mathbf{x}\|} S(\mathbf{x}) + \frac{1 - \cos(\|\mathbf{x}\|)}{\|\mathbf{x}\|^2} S^2(\mathbf{x}) \tag{A2}$$

for $\mathbf{x} \in \mathbb{R}^3$. The logarithm mapping: for $\mathbf{R} \in \mathbb{SO}(3)$

$$\log(\mathbf{R}) = S^{-1}\left(\frac{\theta(\mathbf{R} - \mathbf{R}^\intercal)}{2 \sin\theta}\right) \tag{A3}$$

where $\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right)$. The right Jaocbian: for $\mathbf{x}(\neq \mathbf{0}) \in \mathbb{R}^3$

$$\begin{aligned} \mathrm{J}_r(\mathbf{x}) &= \mathbf{I}_3 - \frac{1 - \cos(\|\mathbf{x}\|)}{\|\mathbf{x}\|^2} S(\mathbf{x}) + \frac{\|\mathbf{x}\| - \sin(\|\mathbf{x}\|)}{\|\mathbf{x}\|^3} S^2(\mathbf{x}), \\ \mathrm{J}_r(\mathbf{0}) &= \mathbf{I}_3 \end{aligned} \tag{A4}$$

References

1. Bachrach, A.; de Winter, A.; He, R.; Hemann, G.; Prentice, S.; Roy, N. RANGE—Robust autonomous navigation in GPS-denied environments. In Proceedings of the International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 1096–1097.
2. Grzonka, S.; Grisetti, G.; Burgard, W. A Fully Autonomous Indoor Quadrotor. *IEEE Trans. Robot.* **2012**, *28*, 90–100.
3. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. Pat. Anal. Mach. Intell.* **2007**, *29*, 1052–1067.
4. Civera, J.; Davison, A.J.; Montiel, J.M.M. Inverse Depth Parametrization for Monocular SLAM. *IEEE Trans. Robot.* **2008**, *24*, 932–945.
5. Eade, E.; Drummond, T. Unified Loop Closing and Recovery for Real Time Monocular SLAM. *BMVC* **2008**, *13*, 136.
6. Eade, E.; Drummond, T. Scalable Monocular SLAM. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 17–22 June 2006; pp. 469–476.
7. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
8. Engel, J.; Sturm, J.; Cremers, D. Semi-dense Visual Odometry for a Monocular Camera. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, 1–8 December 2013; pp. 1449–1456.
9. Olson, C.F.; Matthies, L.H.; Schoppers, M.; Maimone, M.W. Stereo ego-motion improvements for robust rover navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea, 21–26 May 2001; pp. 1099–1104.
10. Se, S.; Lowe, D.; Little, J. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *Int. J. Robot. Res.* **2002**, *21*, 735–758.
11. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Robot. Res.* **2015**, *34*, 314–334.
12. Izadi, S.; Kim, D.; Hilliges, O.; Molyneaux, D.; Newcombe, R.; Kohli, P.; Shotton, J.; Hodges, S.; Freeman, D.; Davison, A.; et al. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, 16–19 October 2011; pp. 559–568.
13. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D mapping with an RGB-D camera. *IEEE Trans. Robot.* **2014**, *30*, 177–187.
14. Scaramuzza, D.; Siegwart, R. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *IEEE Trans. Robot.* **2008**, *24*, 1015–1026.
15. Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. In Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–7 June 2014; pp. 15–22.
16. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*; Springer International Publishing: Cham, Switzerland, 2014; pp. 834–849.
17. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pat. Anal. Mach. Intell.* **2017**, *PP*, 1.
18. Civera, J.; Davison, A.J.; Montiel, J.M.M. Inverse Depth to Depth Conversion for Monocular SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 2778–2783.
19. Li, M.; Mourikis, A.I. High-precision, consistent EKF-based visual-inertial odometry. *Int. J. Robot. Res.* **2013**, *32*, 690–711.
20. Hesch, J.A.; Kottas, D.G.; Bowman, S.L.; Roumeliotis, S.I. *Observability-Constrained Vision-Aided Inertial Navigation*; Technical Report; University of Minnesota: Minneapolis, MN, USA, 2012.
21. Lynen, S.; Achtelik, M.W.; Weiss, S.; Chli, M.; Siegwart, R. A robust and modular multi-sensor fusion approach applied to MAV navigation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 3923–3929.

22. Jones, E.S.; Soatto, S. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *Int. J. Robot. Res.* **2011**, *30*, 407–430.
23. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163.
24. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. g2o: A general framework for graph optimization. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3607–3613.
25. Agarwal, S.; Mierle, K. Ceres Solver. Available online: http://ceres-solver.org (accessed on 14 November 2017).
26. Kaess, M.; Ranganathan, A.; Dellaert, F. iSAM: Incremental Smoothing and Mapping. *IEEE Trans. Robot.* **2008**, *24*, 1365–1378.
27. Dellaert, F. *Factor graphs and GTSAM: A Hands-on Introduction*; Technical Report; Georgia Institute of Technology: Atlanta, GA, USA, 2012.
28. Sibley, G.; Matthies, L.; Sukhatme, G. A sliding window filter for incremental SLAM. In *Unifying Perspectives in Computational and Robot Vision*; Springer: Berlin, Germany, 2008; pp. 103–112.
29. Sibley, G.; Matthies, L.; Sukhatme, G. Sliding window filter with application to planetary landing. *J. Field Robot.* **2010**, *27*, 587–608.
30. Qin, T.; Shen, S. Robust initialization of monocular visual-inertial estimation on aerial robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017.
31. Weiss, S.; Achtelik, M.W.; Lynen, S.; Achtelik, M.C.; Kneip, L.; Chli, M.; Siegwart, R. Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium. *J. Field Robot.* **2013**, *30*, 803–831.
32. Roumeliotis, S.I.; Burdick, J.W. Stochastic cloning: A generalized framework for processing relative state measurements. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; pp. 1788–1795.
33. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21.
34. Concha, A.; Loianno, G.; Kumar, V.; Civera, J. Visual-inertial direct SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 1331–1338.
35. Zhang, T.; Wu, K.; Su, D.; Huang, S.; Dissanayake, G. An Invariant-EKF VINS Algorithm for Improving Consistency. *arXiv* **2017**, arXiv:1702.07920.
36. Hesch, J.A.; Kottas, D.G.; Bowman, S.L.; Roumeliotis, S.I. Consistency Analysis and Improvement of Vision-aided Inertial Navigation. *IEEE Trans. Robot.* **2014**, *30*, 158–176.
37. Zhang, T.; Wu, K.; Song, J.; Huang, S.; Dissanayake, G. Convergence and Consistency Analysis for a 3-D Invariant-EKF SLAM. *IEEE Robot. Autom. Lett.* **2017**, *2*, 733–740.
38. Lupton, T.; Sukkarieh, S. Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions. *IEEE Trans. Robot.* **2012**, *28*, 61–76.
39. Shen, S.; Mulgaonkar, Y.; Michael, N.; Kumar, V. Initialization-free monocular visual-inertial state estimation with application to autonomous MAVs. In *Experimental Robotics*; Springer International Publishing: Cham, Switzerland, 2016; pp. 211–227.
40. Mur-Artal, R.; Tardós, J.D. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robot. Autom. Lett.* **2017**, *2*, 796–803.
41. Carlone, L.; Karaman, S. Attention and anticipation in fast visual-inertial navigation. In Proceedings of the International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 3886–3893.
42. Joshi, S.; Boyd, S. Sensor Selection via Convex Optimization. *IEEE Trans. Signal Process.* **2009**, *57*, 451–462.
43. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163.
44. Shi, J.; Tomasi, C. Good features to track. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June 1994; pp. 593–600.
45. Eigen Is a C++ Template Library for Linear Algebra: Matrices, Vectors, Numerical Solvers, and Related Algorithms. Available online: http://eigen.tuxfamily.org (accessed on 14 November 2017).

46. Furgale, P.; Rehder, J.; Siegwart, R. Unified temporal and spatial calibration for multi-sensor systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1280–1286.
47. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 16–21 June 2012; pp. 3354–3361.
48. Barfoot, T.D. *State Estimation for Robotics*; Cambridge University Press: Cambridge, UK, 2017.