Article

PL-VIO: Tightly-Coupled Monocular Visual–Inertial Odometry Using Point and Line Features

Yijia He [1,2,*] , **Ji Zhao [3]**, **Yue Guo [1,2]**, **Wenhao He [1]** and **Kui Yuan [1]**

[1]   Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China;
      guoyue2013@ia.ac.cn (Y.G.); wenhao.he@ia.ac.cn (W.H.); kui.yuan@ia.ac.cn (K.Y.)
[2]   University of Chinese Academy of Sciences, Beijing 100049, China
[3]   The ReadSense Ltd., Shanghai 200040, China; zhaoji84@gmail.com
[*]   Correspondence: heyijia2013@ia.ac.cn; Tel.: +86-186-1387-5058

check for
updates

Abstract: To address the problem of estimating camera trajectory and to build a structural three-dimensional (3D) map based on inertial measurements and visual observations, this paper proposes point–line visual–inertial odometry (PL-VIO), a tightly-coupled monocular visual–inertial odometry system exploiting both point and line features. Compared with point features, lines provide significantly more geometrical structure information on the environment. To obtain both computation simplicity and representational compactness of a 3D spatial line, Plücker coordinates and orthonormal representation for the line are employed. To tightly and efficiently fuse the information from inertial measurement units (IMUs) and visual sensors, we optimize the states by minimizing a cost function which combines the pre-integrated IMU error term together with the point and line re-projection error terms in a sliding window optimization framework. The experiments evaluated on public datasets demonstrate that the PL-VIO method that combines point and line features outperforms several state-of-the-art VIO systems which use point features only.

Keywords: sensor fusion; visual–inertial odometry; tightly-coupled; point and line features

1. Introduction

Localization and navigation have attracted much attention in recent years with respect to a wide range of applications, particularly for self-driving cars, service robots, and unmanned aerial vehicles, etc. Several types of sensors are utilized for localization and navigation, such as global navigation satellite systems (GNSSs) [1], laser lidar [2,3], inertial measurement units (IMUs), and cameras [4,5]. However, they have obvious respective drawbacks: GNSSs only provide reliable localization information if there is a clear sky view [6]; laser lidar suffers from a reflection problem for objects with glass surfaces [7]; measurements from civilian IMUs are noisy, such that inertial navigation systems may drift quickly due to error accumulation [8]; and monocular simultaneous localization and mapping (SLAM) can only recover the motion trajectory up to a certain scale and it tends to be lost when the camera moves fast or illumination changes dramatically [9–11]. As a result, sensor fusion methods, especially for visual–inertial navigation systems, have drawn widespread attention [12]. The acceleration and angular velocity information from an IMU can significantly improve the monocular SLAM system [13,14]. Furthermore, both IMUs and cameras are light-weight and low-cost, and as such they are widely used in civilian applications.

According to directly or indirectly fused measurements from sensors, visual–inertial odometry (VIO) systems can be divided into two main streams: loosely-coupled and tightly-coupled approaches. Loosely-coupled approaches [15,16] process images and IMU measurements by two estimators that estimate relative motion separately and fuse the estimates from two estimators to obtain the final

result. Tightly-coupled approaches [17,18] use one estimator to find optimal estimates by fusing raw measurements from the camera and IMU directly. Compared to loosely-coupled approaches, tightly-coupled approaches are generally more accurate and robust. In this paper, the proposed PL-VIO method is a tightly-coupled VIO system. Related works on tightly-coupled VIO approaches can be categorized by the number of linearizations in the measurement model [14]. These approaches based on the extended Kalman filter (EKF) process a measurement only once in the updating step, while batch nonlinear optimization approaches linearize multiple times during the optimization step. Filtering-based approaches [19,20] integrate IMU measurements to propagate/predict the state, and then update/correct the latest state with visual measurements. Since the coordinates of three-dimensional (3D) landmarks are included in the state vector, the computational complexity of the EKF increases quadratically with the number of landmarks. To address this problem, Mourikis and Roumeliotis [21] proposed the multi-state constraint Kalman filter (MSCKF) which marginalizes out the landmark coordinates from the state vector. A drawback of this method is that the landmark measurements used to update the state need to be moved out of view of the camera, which means that not all the current visual measurements are used in the filter. Furthermore, the linearization errors make the filter inconsistent [14].

Optimization-based approaches obtain the optimal estimate by minimizing a joint nonlinear cost function with IMU measurement residuals and visual re-projection residuals. Thus, optimization-based approaches can repeat the linearization of a state vector at different points to achieve higher accuracy than filtering-based methods [14]. The IMU measurement constraints are computed by integrating IMU measurements between frames. However, the standard IMU integration method is closely connected with the initial IMU body state at the first frame. When the estimated state changes, all integrated IMU measurements need to be re-calculated. Lupton and Sukkarieh [22] proposed an IMU pre-integration technology which avoids such repeated integrations. IMU pre-integration has been widely used in optimization-based VIO [18,23,24]. Forster et al. [14] reformulated the IMU pre-integration by treating the rotation group on a manifold instead of using Euler angles. Liu et al. [13] proposed the continuous pre-integration method. Although optimization-based approaches have achieved high accuracy, computation becomes expensive with more and more landmarks being added into the optimization. OKVIS [18] used a first-in-last-out sliding window method for bound computation by marginalizing the measurements from the oldest state. Shen et al. [23] proposed a two-way marginalization to selectively marginalize the body state and landmarks.

Although significant achievements have been made in the VIO area, most VIO systems only use the point features as the visual information. However, point detection in textureless environments and point tracking in illumination-changing scenes are challenging [25,26]. In contrast, line segments are a proper alternative solution in these scenes. Additionally, line segments provide more structural information on the environment than points [27]. For visual-only SLAM, there are several works combining point and line features to estimate camera motion [28,29]. The simplest way to integrate line features in a SLAM system is to use two endpoints to represent the line. Matching the endpoints of a line from different views is difficult. Furthermore, the 3D spatial line only has four degrees-of-freedom (DoFs), while two 3D endpoints introduce six parameters, which results in over-parameterization. Bartoli and Sturm [30] proposed the orthonormal representation, which uses a three-DoF rotation matrix and a one-DoF rotation matrix to update the line parameters during optimization. The orthonormal representation has been used in some stereo visual SLAM systems [27,31]. For VIO approaches, Kottas and Roumeliotis [26] investigated the observability of the VIO using line features only. Kong et al. [25] built a stereo VIO system combining point and line features by utilizing trifocal geometry. However, these works involve filtering-based VIO. In our proposed PL-VIO method, we integrate line features into the optimization framework in order achieve higher accuracy than filtering-based methods.

To build a structural 3D map and obtain the camera's motion, we propose the PL-VIO system, which optimizes the system states by jointly minimizing the IMU pre-integration constraints together with the point and line re-projection errors in sliding windows. Compared to the traditional methods which only use point features, our method utilizes the additional line feature, aiming to increase the robustness and accuracy in an illumination-changing environment. Our main contributions are as follows:

- To the best of our knowledge, the proposed PL-VIO is the first optimization-based monocular VIO system using both points and lines as landmarks.
- To tightly and efficiently fuse the information from visual and inertial sensors, we introduce a sliding window model with IMU pre-integration constraints and point/line features. To represent a 3D spatial line compactly in optimization, the orthonormal representation for a line is employed. All the Jacobian matrices of error terms with respect to IMU body states are derived for solving the sliding window optimization efficiently.
- We compare the performances of the proposed PL-VIO with three state-of-the-art monocular VIO methods including ROVIO [17], OKVIS [18], and VINS-Mono [32] on both the EuRoc dataset and the PennCOSYVIO dataset, for which detailed evaluation results are reported.

The remainder of this paper is organized as follows. First, we describe the mathematical preliminaries in Section 2, and then formulate the sliding window-based visual–inertial fusion method in Section 3. Next, we describe our PL-VIO system and implementation details in Section 4. Section 5 shows the experimental results. Finally, conclusions and potential future works are given in Section 6.

2. Mathematical Formulation

2.1. Notations

Figure 1 illustrates the visual–inertial sensors, and the visual observations for point and line features. We denote c_i as the camera frame at time $t = i$ and b_i as the IMU body frame at the same time. w is the Earth's inertial frame. $(\cdot)^c$ means the vector $(\cdot)$ is expressed in frame c. Quaternion $\mathbf{q}_{xy}$ is used to rotate a vector from frame y to frame x, and the corresponding matrix form is $\mathbf{R}_{xy}$. We use $\mathbf{p}_{xy}$ to translate a vector from frame y to frame x. Quaternion $\mathbf{q}_{bc}$ and vector $\mathbf{p}_{bc}$ are the extrinsic parameters between the camera frame and the body frame, and these extrinsic parameters are known in the provided datasets or calibrated with the Kalibr calibration toolbox [33]. $\mathbf{f}_j$ and $\mathcal{L}_j$ are the jth point landmark and the line landmark, respectively, in the map. $\mathbf{z}$ represents a measurement; specifically $\mathbf{z}_{\mathbf{f}_j}^{c_i}$ is the jth point feature observed by ith camera frame, and $\mathbf{z}_{b_i b_j}$ represents a pre-integrated IMU measurement between two keyframes.
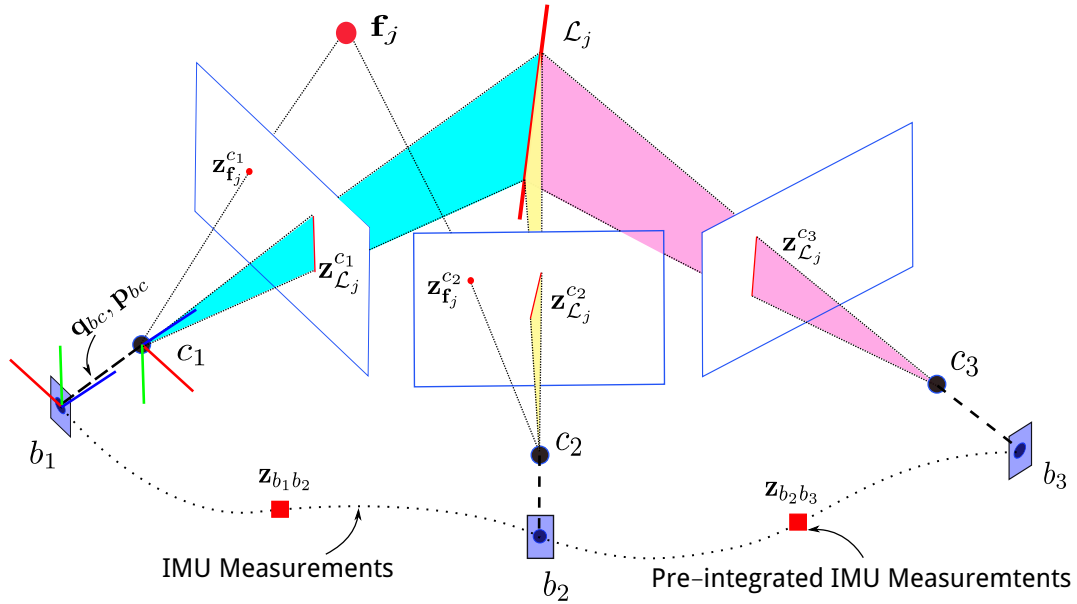
Figure 1. An illustration of visual–inertial sensors, point observations, and line observations. IMU: inertial measurement unit.

2.2. IMU Pre-Integration

A 6-axis IMU, including a 3-axis accelerometer and a 3-axis gyroscope, can measure the acceleration $\mathbf{a}$ and the angular velocity $\boldsymbol{\omega}$ of the body frame with respect to the inertial frame [14]. The raw measurements $\hat{\boldsymbol{\omega}}$ and $\hat{\mathbf{a}}$ are affected by bias and white noise:

$$\hat{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \mathbf{b}_g^b + \mathbf{n}_g^b \tag{1}$$

$$\hat{\mathbf{a}}^b = \mathbf{R}_{bw}(\mathbf{a}^w + \mathbf{g}^w) + \mathbf{b}_a^b + \mathbf{n}_a^b \tag{2}$$

where $\mathbf{b}_g^b$, $\mathbf{b}_a^b$ and $\mathbf{n}_g^b$, $\mathbf{n}_a^b$ are the biases and white noises from gyroscope and accelerometer, respectively. $\mathbf{g}^w = [0, 0, g]^\top$ is the gravity vector in frame w. We use the following kinematics for IMU-driven system [34]:

$$\dot{\mathbf{p}}_{wb_t} = \mathbf{v}_t^w, \qquad \dot{\mathbf{v}}_t^w = \mathbf{a}_t^w, \qquad \dot{\mathbf{q}}_{wb_t} = \mathbf{q}_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}^{b_t} \end{bmatrix} \tag{3}$$

where $\otimes$ denotes the quaternion multiplication operation.

Given the IMU body state at time $t = i$, namely $\mathbf{p}_{wb_i}$, $\mathbf{v}_i^w$, $\mathbf{q}_{wb_j}$, and series values of $\boldsymbol{\omega}$ and $\mathbf{a}$ during the duration $t \in [i, j]$, we can obtain the body state at time $t = j$ by integrating Equation (3):

$$\mathbf{p}_{wb_j} = \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta t + \int\int_{t \in [i,j]} (\mathbf{R}_{wb_t}\mathbf{a}^{b_t} - \mathbf{g}^w)\delta t^2$$

$$\mathbf{v}_j^w = \mathbf{v}_i^w + \int_{t \in [i,j]} (\mathbf{R}_{wb_t}\mathbf{a}^{b_t} - \mathbf{g}^w)\delta t \tag{4}$$

$$\mathbf{q}_{wb_j} = \int_{t \in [i,j]} \mathbf{q}_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}^{b_t} \end{bmatrix} \delta t$$

where Δt is the time difference between i and j. In Equation (4), the body state propagation starts from the ith frame b_i. When the state of b_i is changed, we need to re-propagate all the measurements. Since body states are adjusted at each iteration during the optimization, Equation (4) is time-consuming. By decomposing $\mathbf{q}_{wb_j}$ to $\mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_i b_t}$, Equation (4) can be written as:

$$\mathbf{p}_{wb_j} = \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta t - \frac{1}{2}\mathbf{g}^w \Delta t^2 + \mathbf{R}_{wb_i}\boldsymbol{\alpha}_{b_i b_j}$$

$$\mathbf{v}_j^w = \mathbf{v}_i^w - \mathbf{g}^w \Delta t + \mathbf{R}_{wb_i}\boldsymbol{\beta}_{b_i b_j} \tag{5}$$

$$\mathbf{q}_{wb_j} = \mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_i b_j}$$

where

$$\boldsymbol{\alpha}_{b_i b_j} = \int\int_{t \in [i,j]} (\mathbf{R}_{b_i b_t}\mathbf{a}^{b_t})\delta t^2$$

$$\boldsymbol{\beta}_{b_i b_j} = \int_{t \in [i,j]} (\mathbf{R}_{b_i b_t}\mathbf{a}^{b_t})\delta t \tag{6}$$

$$\mathbf{q}_{b_i b_j} = \int_{t \in [i,j]} \mathbf{q}_{b_i b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\boldsymbol{\omega}^{b_t} \end{bmatrix} \delta t$$

$\mathbf{z}_{b_i b_j} = [\boldsymbol{\alpha}_{b_i b_j}, \boldsymbol{\beta}_{b_i b_j}, \mathbf{q}_{b_i b_j}]^\top$ are called pre-integration measurements [22] and can be calculated directly without knowing the body states of b_i, which means when body state is changed the re-propagation is not necessary. We treat the pre-integrated measurements as constraint factors between successive keyframes.

The pre-integration model (Equation (6)) is derived from continuous time and neglects the biases and noises. In practice, IMU measurements are collected from discrete times, and noise should be considered. In this work, we use mid-point integration to integrate the IMU measurements. The IMU body propagation using measurements at discrete moments k and $k + 1$ is calculated by:

$$\hat{\boldsymbol{\omega}} = \frac{1}{2}((\hat{\boldsymbol{\omega}}^{b_k} - \mathbf{b}_g^{b_k} + \mathbf{n}_g^{b_k}) + (\hat{\boldsymbol{\omega}}^{b_{k+1}} - \mathbf{b}_g^{b_k} + \mathbf{n}_g^{b_{k+1}}))$$

$$\hat{\mathbf{q}}_{b_i b_{k+1}} = \hat{\mathbf{q}}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\hat{\boldsymbol{\omega}}\delta t \end{bmatrix}$$

$$\hat{\mathbf{a}} = \frac{1}{2}(\mathbf{R}_{b_i b_k}(\hat{\mathbf{a}}^{b_k} - \mathbf{b}_a^{b_k} + \mathbf{n}_a^{b_k}) + \mathbf{R}_{b_i b_{k+1}}(\hat{\mathbf{a}}^{b_{k+1}} - \mathbf{b}_a^{b_{k+1}} + \mathbf{n}_g^{b_{k+1}})) \tag{7}$$

$$\hat{\boldsymbol{\alpha}}_{b_i b_{k+1}} = \hat{\boldsymbol{\alpha}}_{b_i b_k} + \hat{\boldsymbol{\beta}}_{b_i b_k}\delta t + \frac{1}{2}\hat{\mathbf{a}}\delta t^2$$

$$\hat{\boldsymbol{\beta}}_{b_i b_{k+1}} = \hat{\boldsymbol{\beta}}_{b_i b_k} + \hat{\mathbf{a}}\delta t$$

At the beginning, $k = i$, we have $\mathbf{q}_{b_i b_i} = [0, 0, 0, 1]^\top$, and $\boldsymbol{\alpha}_{b_i b_i}, \boldsymbol{\beta}_{b_i b_i}$ are zero vectors. In Equation (7), in order to compute the pre-integration measurements efficiently, we assume biases are constant between two keyframes:

$$\mathbf{b}_a^{b_k} = \mathbf{b}_a^{b_{k+1}}, \quad \mathbf{b}_g^{b_k} = \mathbf{b}_g^{b_{k+1}}, \quad k \in [i, j-1] \tag{8}$$

In practice, biases change slowly. We model biases with random walk noises:

$$\mathbf{b}_a^{b_{k+1}} = \mathbf{b}_a^{b_k} + \mathbf{n}_{\mathbf{b}_a}\delta t, \quad \mathbf{b}_g^{b_{k+1}} = \mathbf{b}_g^{b_k} + \mathbf{n}_{\mathbf{b}_g}\delta t \tag{9}$$

where the Gaussian white noises are defined as $\mathbf{n}_{\mathbf{b}_a} \in \mathcal{N}(0, \sigma_{b_a}^2)$ and $\mathbf{n}_{\mathbf{b}_g} \in \mathcal{N}(0, \sigma_{b_g}^2)$. When bias changes with a small increments, instead of computing pre-integrated measurements iteratively, we use a first-order approximation to update $\hat{\mathbf{q}}_{b_i b_j}, \hat{\boldsymbol{\alpha}}_{b_i b_j}, \hat{\boldsymbol{\beta}}_{b_i b_j}$ [14]:

$$\hat{\boldsymbol{\alpha}}_{b_i b_j} \leftarrow \hat{\boldsymbol{\alpha}}_{b_i b_j} + \mathbf{J}^{\alpha}_{b_a^i} \delta \mathbf{b}^{b_i}_a + \mathbf{J}^{\alpha}_{b_g^i} \delta \mathbf{b}^{b_i}_g$$

$$\hat{\boldsymbol{\beta}}_{b_i b_j} \leftarrow \hat{\boldsymbol{\beta}}_{b_i b_j} + \mathbf{J}^{\beta}_{b_a^i} \delta \mathbf{b}^{b_i}_a + \mathbf{J}^{\beta}_{b_g^i} \delta \mathbf{b}^{b_i}_g \tag{10}$$

$$\hat{\mathbf{q}}_{b_i b_j} \leftarrow \hat{\mathbf{q}}_{b_i b_j} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{J}^{q}_{b_g^i} \delta \mathbf{b}^{b_i}_g \end{bmatrix}$$

where $\mathbf{J}^{\alpha}_{b_a^i} = \frac{\partial \boldsymbol{\alpha}_{b_i b_j}}{\partial \delta \mathbf{b}^{b_i}_a}$, $\mathbf{J}^{\alpha}_{b_g^i} = \frac{\partial \boldsymbol{\alpha}_{b_i b_j}}{\partial \delta \mathbf{b}^{b_i}_g}$, $\mathbf{J}^{\beta}_{b_a^i} = \frac{\partial \boldsymbol{\beta}_{b_i b_j}}{\partial \delta \mathbf{b}^{b_i}_a}$, $\mathbf{J}^{\beta}_{b_g^i} = \frac{\partial \boldsymbol{\beta}_{b_i b_j}}{\partial \delta \mathbf{b}^{b_i}_g}$, $\mathbf{J}^{q}_{b_g^i} = \frac{\partial \mathbf{q}_{b_i b_j}}{\partial \delta \mathbf{b}^{b_i}_g}$ are the Jacobian matrices of pre-integrated measurements with respect to bias at time *i*. They can be derived with error state transformation matrices, as shown in Appendix A. The covariance matrix of pre-integrated measurements $\boldsymbol{\Sigma}_{b_i b_j}$ can be computed iteratively with IMU propagation, and more details are provided in Appendix A.

2.3. Geometric Representation of Line

A straight line only has four DoFs. Thus, the compact parameterization of a straight line is with four parameters. In our system, we treat a straight line in 3D space as an infinite line and adopt two parameterizations for a 3D line as in [27]. Plücker line coordinates consisting of six parameters are used for transformation and projection due to their simplicity. An orthonormal representation consisting of four parameters is used for optimization due to its compactness.

2.3.1. Plücker Line Coordinates

In Figure 2a, a 3D spatial line $\mathcal{L}$ in Plücker coordinates is represented by $\mathcal{L} = (\mathbf{n}^{\top}, \mathbf{d}^{\top})^{\top} \in \mathbb{R}^6$, where $\mathbf{d} \in \mathbb{R}^3$ is the line direction vector, and $\mathbf{n} \in \mathbb{R}^3$ is the normal vector of the plane determined by the line and the coordinate origin. The Plücker coordinates are over-parameterized since there is an implicit constraint between the vector $\mathbf{n}$ and $\mathbf{d}$, i.e., $\mathbf{n}^{\top} \mathbf{d} = 0$. Therefore, the Plücker coordinates can not be directly used in unconstrained optimization. However, with a 3D line represented by a normal vector and a direction vector it is simple to perform triangulation from two views geometrically, and it is also convenient to model the line geometry transformation.
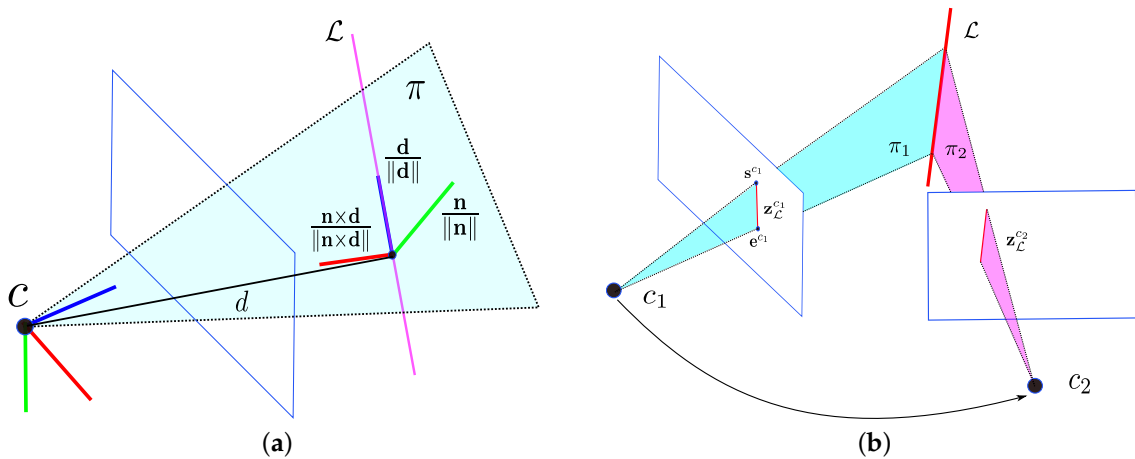

Figure 2. Plücker coordinates for line features. (**a**) Plücker line coordinates; (**b**) Initialization of a newly observed line.

For line geometry transformation, given the transformation matrix $\mathbf{T}_{cw} = \begin{bmatrix} \mathbf{R}_{cw} & \mathbf{p}_{cw} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$ from the world frame w to the camera frame c, we can transform the Plücker coordinates of a line by [30]

$$\mathcal{L}^c = \begin{bmatrix} \mathbf{n}^c \\ \mathbf{d}^c \end{bmatrix} = \mathcal{T}_{cw}\mathcal{L}_w = \begin{bmatrix} \mathbf{R}_{cw} & [\mathbf{p}_{cw}]_\times \mathbf{R}_{cw} \\ \mathbf{0} & \mathbf{R}_{cw} \end{bmatrix} \mathcal{L}^w \tag{11}$$

where $[\cdot]_\times$ is the skew-symmetric matrix of a three-dimensional vector, and $\mathcal{T}_{cw}$ is the transform matrix used to transform a line from frame w to frame c.

When a new line landmark is observed in two different camera views, the Plücker coordinates are easily calculated. As shown in Figure 2b, the 3D line $\mathcal{L}$ is captured by cameras c_1 and c_2 as $\mathbf{z}_\mathcal{L}^{c_1}$ and $\mathbf{z}_\mathcal{L}^{c_2}$, respectively. The line segment $\mathbf{z}_\mathcal{L}^{c_1}$ in the normalized image plane can be represented by two endpoints, $\mathbf{s}^{c_1} = [u_s, v_s, 1]^\top$ and $\mathbf{e}^{c_1} = [u_e, v_e, 1]^\top$. Three non-collinear points, including two endpoints of a line segment and the coordinate origin $\mathbf{C} = [x_0, y_0, z_0]^\top$, determine a plane $\boldsymbol{\pi} = [\pi_x, \pi_y, \pi_z, \pi_w]^\top$ in 3D space:

$$\pi_x(x - x_0) + \pi_y(y - y_0) + \pi_z(z - z_0) = 0 \tag{12}$$

where

$$\begin{bmatrix} \pi_x \\ \pi_y \\ \pi_z \end{bmatrix} = [\mathbf{s}^{c_1}]_\times \mathbf{e}^{c_1}, \quad \pi_w = \pi_x x_0 + \pi_y y_0 + \pi_z z_0 \tag{13}$$

Given the two plane $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ in camera frame c_1, the dual Plücker matrix $\mathbf{L}^*$ can be computed by

$$\mathbf{L}^* = \begin{bmatrix} [\mathbf{d}]_\times & \mathbf{n} \\ -\mathbf{n}^\top & 0 \end{bmatrix} = \boldsymbol{\pi}_1 \boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2 \boldsymbol{\pi}_1^\top \in \mathbb{R}^{4\times 4} \tag{14}$$

The Plücker coordinates $\mathcal{L}$ in camera frame c_1 are easily extracted from the dual Plücker matrix $\mathbf{L}^*$. It can be seen that $\mathbf{n}$ and $\mathbf{d}$ do not need to be unit vectors.

2.3.2. Orthonormal Representation

Since 3D spatial lines only have four DoFs, the orthonormal representation $(\mathbf{U}, \mathbf{W}) \in SO(3) \times SO(2)$ is more suitable than Plücker coordinates during optimization. Additionally, the orthonormal representation and Plücker coordinates can be converted from each other, which means we can adopt both of them in a SLAM system for different purposes. In this section, we will introduce the details of orthonormal representation. As shown in Figure 2a, a coordinate system is defined on the 3D line. The normalized normal vector and the normalized direction vector are the two axes of the coordinate system. The third axis is determined by crossing the other two axes vectors. We can define the rotation matrix between the line coordinate and the camera frame as $\mathbf{U}$:

$$\mathbf{U} = \mathbf{R}\left(\boldsymbol{\psi}\right) = \begin{bmatrix} \frac{\mathbf{n}}{\|\mathbf{n}\|} & \frac{\mathbf{d}}{\|\mathbf{d}\|} & \frac{\mathbf{n}\times\mathbf{d}}{\|\mathbf{n}\times\mathbf{d}\|} \end{bmatrix} \tag{15}$$

where $\boldsymbol{\psi} = [\psi_1, \psi_2, \psi_3]^\top$ consists of the rotation angles around the x-, y-, and z-axes of a camera frame. The relationship between the Plücker coordinates and $\mathbf{U}$ is:

$$\begin{bmatrix} \mathbf{n} & \mathbf{d} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{n}}{\|\mathbf{n}\|} & \frac{\mathbf{d}}{\|\mathbf{d}\|} & \frac{\mathbf{n}\times\mathbf{d}}{\|\mathbf{n}\times\mathbf{d}\|} \end{bmatrix} \begin{bmatrix} \|\mathbf{n}\| & 0 \\ 0 & \|\mathbf{d}\| \\ 0 & 0 \end{bmatrix} \tag{16}$$

Since the combination of $(\|\mathbf{n}\|, \|\mathbf{d}\|)$ in Equation (16) only has one DoF, we can use trigonometric functions to represent it:

$$\mathbf{W} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} = \frac{1}{\sqrt{(\|\mathbf{n}\|^2 + \|\mathbf{d}\|^2)}} \begin{bmatrix} \|\mathbf{n}\| & -\|\mathbf{d}\| \\ \|\mathbf{d}\| & \|\mathbf{n}\| \end{bmatrix} \tag{17}$$

where ϕ is a rotation angle. Recall that the distance from coordinate origin to the 3D line is $d = \frac{\|\mathbf{n}\|}{\|\mathbf{d}\|}$, so $\mathbf{W}$ contains the information about the distance d. According to the definitions of $\mathbf{U}$ and $\mathbf{W}$, these four DoFs include three DoFs from the rotation matrix, which transforms the line coordinate to the camera frame, and one DoF from the distance d. We use $\mathcal{O} = [\boldsymbol{\psi}, \phi]^\top$ as the minimal representation of a 3D spatial line during optimization.

Once a 3D line $\mathcal{L}$ has been optimized with the orthonormal representation, the corresponding Plücker coordinates for the line can be computed by:

$$\mathcal{L}' = [w_1 \mathbf{u}_1^T, w_2 \mathbf{u}_2^T]^T = \frac{1}{\sqrt{(\|\mathbf{n}\|^2 + \|\mathbf{d}\|^2)}} \mathcal{L} \tag{18}$$

where $\mathbf{u}_i$ is the ith column of matrix $\mathbf{U}$, $w_1 = \cos(\phi)$, and $w_2 = \sin(\phi)$. There is a scale factor between $\mathcal{L}'$ and $\mathcal{L}$, but they represent the same 3D spatial line.

3. Tightly-Coupled Visual–Inertial Fusion

In visual SLAM, bundle adjustment is used to optimize the camera poses and 3D map points by minimizing the re-projection error in image planes. Bundle adjustment by nonlinear optimization can be treated as a factor graph [35] as shown in Figure 3a: round nodes are the camera poses or 3D landmarks needed to be optimized; edges with square boxes represent the visual measurements as constraints between nodes. For visual–inertial fusion, we can also use the tightly-coupled graph-based framework to optimize all the state variables of the visual–inertial system [23]. As shown in Figure 3b, the factor graph not only contains the visual measurements, but also takes the pre-integrated IMU measurements as edges to constrain the successive IMU body states.
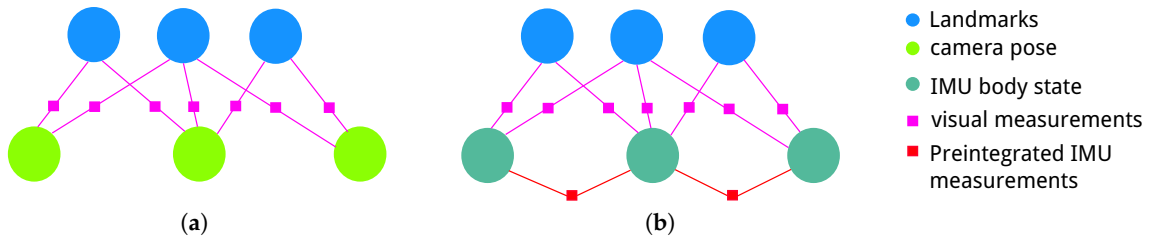

●　Landmarks
●　camera pose
●　IMU body state
■　visual measurements
■　Preintegrated IMU measurements

(**a**)　　　　　　　　　　　(**b**)

Figure 3. Factor graphs of (**a**) the visual simultaneous localization and mapping (SLAM) problem versus (**b**) visual–inertial SLAM.

3.1. Sliding Window Formulation

In order to achieve both computation efficiency and high accuracy, we use the sliding window formulation for factor graph optimization. The full state variables in a sliding window at time i are defined as:

$$\mathcal{X} = [\mathbf{x}_n, \mathbf{x}_{n+1}, ..., \mathbf{x}_{n+N}, \lambda_m, \lambda_{m+1}, ..., \lambda_{m+M}, \mathcal{O}_o, \mathcal{O}_{o+1}, ..., \mathcal{O}_{o+O}]^\top$$
$$\mathbf{x}_i = \left[\mathbf{p}_{wb_i}, \mathbf{q}_{wb_i}, \mathbf{v}_i^w, \mathbf{b}_a^{b_i}, \mathbf{b}_g^{b_i} \right]^\top, i \in [n, n+N] \tag{19}$$

where $\mathbf{x}_i$ is described by the ith IMU body position, velocity, and orientation in the world frame, and biases in the IMU body frame. Subscripts n, m, and o are the start indexes of body states, point landmarks, and line landmarks, respectively. N is the number of keyframes in the sliding window.

M and O are the numbers of point landmarks and line landmarks observed by all keyframes in the sliding window, respectively. We only use one variable, the inverse depth λ_k, to parameterize the kth point landmark from its first observed keyframe. $\mathcal{O}_l$ is the orthonormal representation of the lth line feature in the world frame w.

We optimize all the state variables in the sliding window by minimizing the sum of cost terms from all the measurement residuals:

$$
\min_{\mathcal{X}} \rho \left(\left\| \mathbf{r}_p - \mathbf{J}_p \mathcal{X} \right\|_{\Sigma_p}^2 \right) + \sum_{i \in B} \rho \left(\left\| \mathbf{r}_b(\mathbf{z}_{b_i b_{i+1}}, \mathcal{X}) \right\|_{\Sigma_{b_i b_{i+1}}}^2 \right)
$$
$$
+ \sum_{(i,j) \in F} \rho \left(\left\| \mathbf{r}_f(\mathbf{z}_{\mathbf{f}_j}^{c_i}, \mathcal{X}) \right\|_{\Sigma_{\mathbf{f}_j}^{c_i}}^2 \right) + \sum_{(i,l) \in L} \rho \left(\left\| \mathbf{r}_l(\mathbf{z}_{\mathcal{L}_i}^{c_i}, \mathcal{X}) \right\|_{\Sigma_{\mathcal{L}_i}^{c_i}}^2 \right)
\tag{20}
$$

where $\mathbf{r}_b(\mathbf{z}_{b_i b_{i+1}}, \mathcal{X})$ is an IMU measurement residual between the body state $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. B is the set of all pre-integrated IMU measurements in the sliding window. $\mathbf{r}_f(\mathbf{z}_{\mathbf{f}_j}^{c_i}, \mathcal{X})$ and $\mathbf{r}_l(\mathbf{z}_{\mathcal{L}_i}^{c_i}, \mathcal{X})$ are the point feature re-projection residual and the line feature re-projection residual, respectively. F and L are the sets of point features and line features observed by camera frames. $\{\mathbf{r}_p, \mathbf{J}_p\}$ is prior information that can be computed after marginalizing out a frame in the sliding window [23], and $\mathbf{J}_p$ is the prior Jacobian matrix from the resulting Hessian matrix after the previous optimization. ρ is the Cauchy robust function used to suppress outliers.

We use Levenberg–Marquard algorithm to solve the nonlinear optimization problem (20). The optimal state estimates $\mathcal{X}$ can be found by iteratively update from an initial guess $\mathcal{X}_0$ as:

$$
\mathcal{X}'_{t+1} = \mathcal{X}_t \oplus \delta \mathcal{X}
\tag{21}
$$

where $\oplus$ is the operator used to update parameters with increment $\delta \mathcal{X}$. For position, velocity, bias, and inverse depth, the update operator and increments δ are easily defined:

$$
\mathbf{p}' = \mathbf{p} + \delta \mathbf{p}, \quad \mathbf{v}' = \mathbf{v} + \delta \mathbf{v}, \quad \lambda' = \lambda + \delta \lambda, \quad \mathbf{b}' = \mathbf{b} + \delta \mathbf{b}
\tag{22}
$$

However, the update operator and increments δ for state attitude $\mathbf{q}$ are more complicated. Four parameters are used in quaternion to represent the three-DoF rotation, so it is over-parameterized. The increment for rotation should only be three-dimensional. Similar to [18], we use a perturbation $\delta \boldsymbol{\theta} \in \mathbb{R}^3$ in the tangent space as the rotation increment. Thus, rotation $\mathbf{q}$ can be updated by the quaternion multiplication:

$$
\mathbf{q}' = \mathbf{q} \otimes \delta \mathbf{q}, \quad \delta \mathbf{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta} \end{bmatrix}
\tag{23}
$$

We can also write it as a rotation matrix form:

$$
\mathbf{R}' \approx \mathbf{R}(\mathbf{I} + [\delta \boldsymbol{\theta}]_\times)
\tag{24}
$$

where $\mathbf{I}$ is a 3×3 identity matrix. Similarly, we can update the orthonormal representation as:

$$
\mathbf{U}' \approx \mathbf{U}(\mathbf{I} + [\delta \boldsymbol{\psi}]_\times)
$$
$$
\mathbf{W}' \approx \mathbf{W} \left(\mathbf{I} + \begin{bmatrix} 0 & -\delta \phi \\ \delta \phi & 0 \end{bmatrix} \right)
\tag{25}
$$

The increment for the orthonormal representation is $\delta \mathcal{O} = [[\delta \boldsymbol{\psi}]_\times, \delta \phi]^\top$. Finally, the increment $\delta \mathcal{X}$ during the optimization can be defined as:

$$\delta\mathcal{X} = [\delta\mathbf{x}_n, \delta\mathbf{x}_{n+1}, ..., \delta\mathbf{x}_{n+N}, \delta\lambda_m, \delta\lambda_{m+1}, ..., \delta\lambda_{m+M}, \delta\mathcal{O}_o, \delta\mathcal{O}_{o+1}, ..., \delta\mathcal{O}_{o+O}]^\top$$
$$\delta\mathbf{x}_i = \left[\delta\mathbf{p}, \delta\boldsymbol{\theta}, \delta\mathbf{v}, \delta\mathbf{b}_a^{b_i}, \delta\mathbf{b}_g^{b_i}\right]^\top, i \in [n, n+N]$$

(26)

At each iteration, the increment $\delta\mathcal{X}$ can be solved by Equation (20):

$$(\mathbf{H}_p + \mathbf{H}_b + \mathbf{H}_f + \mathbf{H}_l)\delta\mathcal{X} = (\mathbf{b}_p + \mathbf{b}_b + \mathbf{b}_f + \mathbf{b}_l)$$

(27)

where $\mathbf{H}_p, \mathbf{H}_b, \mathbf{H}_f$, and $\mathbf{H}_l$ are the Hessian matrices for prior residuals, IMU measurement residuals, and point and line re-projection residuals, respectively. For residual $\mathbf{r}_{(\cdot)}$, we have $\mathbf{H}_{(\cdot)} = \mathbf{J}_{(\cdot)}^\top \Sigma_{(\cdot)}^{-1} \mathbf{J}_{(\cdot)}$ and $\mathbf{b}_{(\cdot)} = -\mathbf{J}_{(\cdot)}^\top \Sigma_{(\cdot)}^{-1} \mathbf{r}_{(\cdot)}$, where $\mathbf{J}_{(\cdot)}$ is the Jacobian matrix of residuals vector $\mathbf{r}_{(\cdot)}$ with respect to $\delta\mathcal{X}$, and $\Sigma_{(\cdot)}$ is the covariance matrix of measurements. Formulations of residuals and Jacobian matrices will be defined in the following subsections.

3.2. IMU Measurement Model

Since the pre-integrated IMU measurement computed by Equation (10) is a constraint factor between two successive keyframes $\mathbf{b}_i$ and $\mathbf{b}_j$, the IMU measurement residual can be defined as:

$$\mathbf{r}_b(\mathbf{z}_{b_i b_j}, \mathcal{X}) = \begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_\theta \\ \mathbf{r}_v \\ \mathbf{r}_{ba} \\ \mathbf{r}_{bg} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2) - \hat{\boldsymbol{\alpha}}_{b_i b_j} \\ 2[\hat{\mathbf{q}}_{b_j b_i} \otimes (\mathbf{q}_{b_i w} \otimes \mathbf{q}_{wb_j})]_{xyz} \\ \mathbf{R}_{b_i w}(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t) - \hat{\boldsymbol{\beta}}_{b_i b_j} \\ \mathbf{b}_a^{b_j} - \mathbf{b}_a^{b_i} \\ \mathbf{b}_g^{b_j} - \mathbf{b}_g^{b_i} \end{bmatrix}_{15 \times 1}$$

(28)

where $[\cdot]_{xyz}$ extracts the real part of a quaternion which is used to approximate the three-dimensional rotation error [18].

During the nonlinear optimization, the Jacobian matrix of the IMU measurement residual with respect to the body state $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed by:

$$\mathbf{J}_b = \begin{bmatrix} \frac{\partial \mathbf{r}_b}{\partial \delta\mathbf{x}_i} & \frac{\partial \mathbf{r}_b}{\partial \delta\mathbf{x}_j} \end{bmatrix}$$

$$\frac{\partial \mathbf{r}_b}{\partial \delta\mathbf{x}_i} = \begin{bmatrix} -\mathbf{R}_{b_i w} & [\mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2)]_\times & -\mathbf{R}_{b_i w}\Delta t & -\mathbf{J}_{b_a^i}^{\boldsymbol{\alpha}} & -\mathbf{J}_{b_g^i}^{\boldsymbol{\alpha}} \\ 0 & \left\lfloor -[\mathbf{q}_{wb_j}^{-1} \otimes \mathbf{q}_{wb_i}]_L [\hat{\mathbf{q}}_{b_i b_j}]_R \right\rfloor_{3\times3} & 0 & 0 & \mathbf{J}_{b_g^i}^{\mathbf{r}_\theta} \\ 0 & [\mathbf{R}_{b_i w}(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)]_\times & -\mathbf{R}_{b_i w} & -\mathbf{J}_{b_a^i}^{\boldsymbol{\beta}} & -\mathbf{J}_{b_g^i}^{\boldsymbol{\beta}} \\ 0 & 0 & 0 & -\mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & -\mathbf{I} \end{bmatrix}_{15\times15}$$

(29)

$$\frac{\partial \mathbf{r}_b}{\partial \delta\mathbf{x}_j} = \begin{bmatrix} -\mathbf{R}_{b_i w} & 0 & 0 & 0 & 0 \\ 0 & \left\lfloor -[\hat{\mathbf{q}}_{b_i b_j}^{-1} \otimes \mathbf{q}_{wb_i}^{-1} \otimes \mathbf{q}_{wb_j}]_L \right\rfloor_{3\times3} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{R}_{b_i w} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I} \end{bmatrix}_{15\times15}$$

where $[\mathbf{q}]_L$ and $[\mathbf{q}]_R$ are the left- and right- quaternion-product matrices, respectively [36]. The operator $\lfloor \cdot \rfloor_{3\times3}$ is used to extract a 3×3 matrix from the bottom right block of $(\cdot)$. The Jacobian matrix is calculated by $\mathbf{J}_{b_g^i}^{\mathbf{r}_\theta} = \frac{\partial \mathbf{r}_\theta}{\partial \mathbf{b}_g^{b_i}} = \left\lfloor -[\mathbf{q}_{wb_j}^{-1} \otimes \mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_i b_j}]_L \right\rfloor_{3\times3} \mathbf{J}_{b_g^i}^{q}$.

3.3. Point Feature Measurement Model

For point features, the re-projection error of a 3D point is the image distance between the projected point and the observed point. In this work, we deal with the point and line feature measurements in the normalized image plane. Given the kth point feature measurement at frame c_j, $\mathbf{z}_{\mathbf{f}_k}^{c_j} = [u_{\mathbf{f}_k}^{c_j}, v_{\mathbf{f}_k}^{c_j}, 1]^\top$, the re-projection error is defined as:

$$
\mathbf{r}_f(\mathbf{z}_{\mathbf{f}_k}^{c_i}, \mathcal{X}) = \begin{bmatrix} \frac{x^{c_j}}{z^{c_j}} - u_{\mathbf{f}_k}^{c_j} \\ \frac{y^{c_j}}{z^{c_j}} - v_{\mathbf{f}_k}^{c_j} \end{bmatrix}
$$

$$
\mathbf{f}_k^{c_j} = \begin{bmatrix} x^{c_j} \\ y^{c_j} \\ z^{c_j} \end{bmatrix} = \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top (\mathbf{R}_{wb_i}((\mathbf{R}_{bc}\frac{1}{\lambda_k} \begin{bmatrix} u_{\mathbf{f}_k}^{c_i} \\ v_{\mathbf{f}_k}^{c_i} \\ 1 \end{bmatrix} + \mathbf{p}_{bc}) + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc}) \tag{30}
$$

where $\mathbf{z}_{\mathbf{f}}^{c_i} = [u_{\mathbf{f}_k}^{c_i}, v_{\mathbf{f}_k}^{c_i}, 1]^\top$ is the first observation of the feature in camera frame c_i, and the inverse depth of the feature λ_k is also defined in camera frame c_i.

In order to minimize the point's re-projection error (30), we need to optimize the rotation and the position of frame b_i, b_j, and the feature inverse depth λ. The corresponding Jacobian matrix can be obtained by the chain rule:

$$
\mathbf{J}_f = \frac{\partial \mathbf{r}_f}{\partial \mathbf{f}^{c_j}} \begin{bmatrix} \frac{\partial \mathbf{f}^{c_j}}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{f}^{c_j}}{\partial \mathbf{x}_j} & \frac{\partial \mathbf{f}^{c_j}}{\partial \delta \lambda} \end{bmatrix} \tag{31}
$$

With

$$
\frac{\partial \mathbf{r}_f}{\partial \mathbf{f}^{c_j}} = \begin{bmatrix} \frac{1}{z^{c_j}} & 0 & -\frac{x^{c_j}}{(z^{c_j})^2} \\ 0 & \frac{1}{z^{c_j}} & -\frac{y^{c_j}}{(z^{c_j})^2} \end{bmatrix}
$$

$$
\frac{\partial \mathbf{f}^{c_j}}{\partial \mathbf{x}_i} = \begin{bmatrix} \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top & -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i}[\mathbf{f}^{b_i}]_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{3\times 15} \tag{32}
$$

$$
\frac{\partial \mathbf{f}^{c_j}}{\partial \mathbf{x}_j} = \begin{bmatrix} -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top & \mathbf{R}_{bc}^T[\mathbf{f}^{b_j}]_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{3\times 15}
$$

$$
\frac{\partial \mathbf{f}^{c_j}}{\partial \delta \lambda} = -\frac{1}{\lambda} \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i}
$$

where $\mathbf{f}^{b_i}$ is the 3D point vector in the ith IMU body frame. We define the covariance matrix of point measurement $\Sigma_{\mathbf{f}_k}^{c_i}$ as a 2×2 diagonal matrix by assuming that the detected point features have pixel noise on both the vertical and horizontal directions in the image plane.

3.4. Line Feature Measurement Model

The re-projection error of a line measurement is defined as the distance from endpoints to the projected line. For a pin-hole model camera, a 3D spatial line $\mathcal{L} = [\mathbf{n}, \mathbf{d}]^\top$ can be projected to the camera image plane by [27]:

$$
\mathbf{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \mathcal{K}\mathbf{n}^c = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix} \mathbf{n}^c \tag{33}
$$

where $\mathcal{K}$ is the projection matrix for a line feature. When projecting a line to the normalized image plane, $\mathcal{K}$ is an identity matrix. From the projection Equation (33), the coordinates of the line segment projected by a 3D line are only related with the normal vector $\mathbf{n}$.

Given a 3D line $\mathcal{L}_l^w$ and the orthonormal presentation $\mathcal{O}_l$ in a world frame, we firstly transform it to camera frame c_i by Equation (11). Then, we project it to the image plane to get the projection line $\mathbf{l}_l^{c_i}$. The re-projection error in camera frame i is defined as

$$\mathbf{r}_l(\mathbf{z}_{\mathcal{L}_l}^{c_i}, \mathcal{X}) = \begin{bmatrix} d(\mathbf{s}_l^{c_i}, \mathbf{l}_l^{c_i}) \\ d(\mathbf{e}_l^{c_i}, \mathbf{l}_l^{c_i}) \end{bmatrix} \tag{34}$$

With $d(\mathbf{s}, \mathbf{l})$ indicating the distance function from endpoint $\mathbf{s}$ to the projection line $\mathbf{l}$:

$$d(\mathbf{s}, \mathbf{l}) = \frac{\mathbf{s}^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \tag{35}$$

The ith body state and lth line parameters are optimized by minimizing the line re-projection error $\mathbf{r}_l(\mathbf{z}_{\mathcal{L}_l}^{c_i})$. The corresponding Jacobian matrix can be obtained by the chain rule:

$$\mathbf{J}_l = \frac{\partial \mathbf{r}_l}{\partial \mathbf{l}^{c_i}} \frac{\partial \mathbf{l}^{c_i}}{\partial \mathcal{L}^{c_i}} \begin{bmatrix} \frac{\partial \mathcal{L}^{c_i}}{\partial \delta \mathbf{x}^i} & \frac{\partial \mathcal{L}^{c_i}}{\partial \mathcal{L}^w} \frac{\partial \mathcal{L}^w}{\partial \delta \mathcal{O}} \end{bmatrix} \tag{36}$$

With

$$
\begin{aligned}
\frac{\partial \mathbf{r}_l}{\partial \mathbf{l}^{c_i}} &= \begin{bmatrix} \frac{-l_1(\mathbf{s}_l^{c_i})^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{u_s}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{-l_2(\mathbf{s}_l^{c_i})^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{v_s}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{1}{(l_1^2+l_2^2)^{(\frac{1}{2})}} \\ \frac{-l_1(\mathbf{e}_l^{c_i})^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{u_e}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{-l_2(\mathbf{e}_l^{c_i})^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{v_e}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{1}{(l_1^2+l_2^2)^{(\frac{1}{2})}} \end{bmatrix}_{2\times3} \\
\frac{\partial \mathbf{l}^{c_i}}{\partial \mathcal{L}^{c_i}} &= \begin{bmatrix} \mathcal{K} & \mathbf{0} \end{bmatrix}_{3\times6} \\
\frac{\partial \mathcal{L}^c}{\partial \delta \mathbf{x}^i} &= \begin{bmatrix} \mathcal{T}_{bc}^{-1} \begin{bmatrix} \mathbf{R}_{wb}^\top [\mathbf{d}^w]_\times \\ \mathbf{0}_{3\times3} \end{bmatrix} & \mathcal{T}_{bc}^{-1} \begin{bmatrix} [\mathbf{R}_{wb}^\top (\mathbf{n}^w + [\mathbf{d}^w]_\times \mathbf{p}_{wb})]_\times \\ [\mathbf{R}_{wb}^\top \mathbf{d}^w]_\times \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{6\times15} \\
\frac{\partial \mathcal{L}^{c_i}}{\partial \mathcal{L}^w} \frac{\partial \mathcal{L}^w}{\partial \delta \mathcal{O}} &= \mathcal{T}_{wc}^{-1} \begin{bmatrix} \mathbf{0} & -w_1 \mathbf{u}_3 & w_1 \mathbf{u}_2 & -w_2 \mathbf{u}_1 \\ w_2 \mathbf{u}_3 & \mathbf{0} & -w_2 \mathbf{u}_1 & w_1 \mathbf{u}_2 \end{bmatrix}_{6\times4}
\end{aligned} \tag{37}
$$

The derivation details are provided in Appendix B. Similar to the point measurement covariance matrix, the covariance matrix of line measurement $\Sigma_{\mathcal{L}_l}^{c_i}$ is defined by assuming the endpoints of a line segment have pixel noise.

4. Monocular Visual Inertial Odometry with Point and Line Features

As shown in Figure 4, the proposed PL-VIO system has two main modules: the front end and the back end. The front-end module is used to pre-process the measurements from IMU and camera. The back-end module is used to estimate and optimize the body states. We will introduce the details in the following subsections.
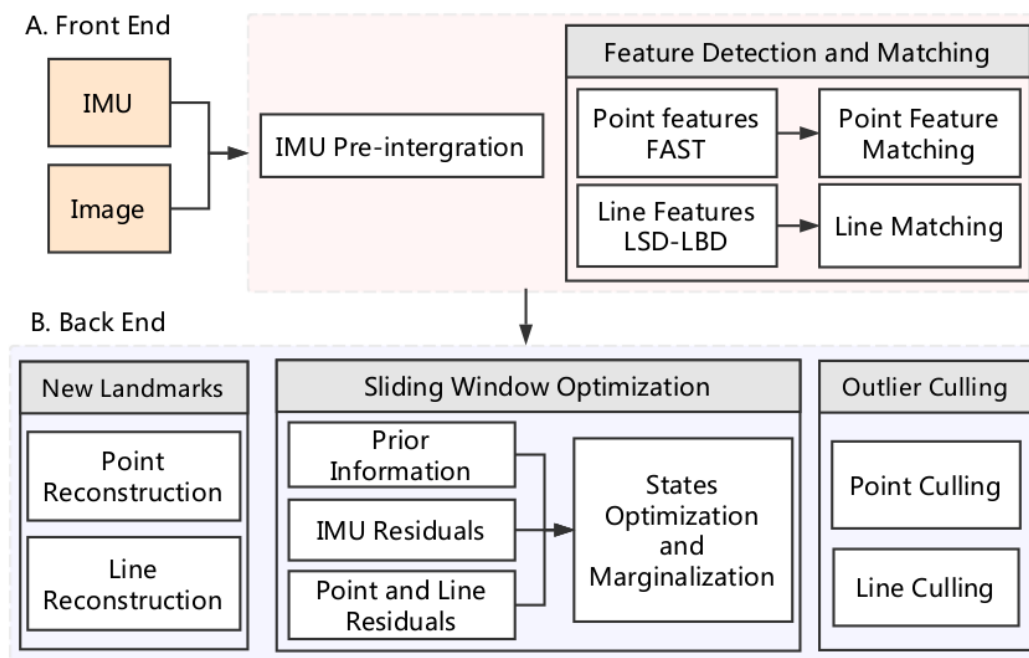
A. Front End

	Feature Detection and Matching	
IMU → IMU Pre-intergration	Point features FAST →	Point Feature Matching
Image →	Line Features LSD-LBD →	Line Matching

B. Back End

New Landmarks	Sliding Window Optimization	Outlier Culling
Point Reconstruction	Prior Information / IMU Residuals / Point and Line Residuals → States Optimization and Marginalization	Point Culling
Line Reconstruction		Line Culling

Figure 4. Overview of our point–line visual–inertial odometry (PL-VIO) system. A is the front end module is used to extract information from the raw measurements; B is the back end module is used to estimate and optimize the body states.

4.1. Front End

The front end extracts information from the raw measurements of the camera and IMU. The body state is updated by propagation with each new IMU measurement, and the newest body state is used as the initial value in sliding window optimization. Additionally, the new IMU measurements are pre-integrated to constrain the successive IMU body states during optimization.

As for image processing, the point and line features are detected in two separate threads. When a new frame comes, the point features are tracked from the previous frame to the new frame by the KLT optical flow algorithm [37]. Then, we use the RANSAC framework with an essential matrix test to remove outliers. If the number of tracked point features is less than a threshold after outlier rejection, new corner features which are detected by the FAST detector [38] will be added. As to the line features, line segments in new frame are detected by the LSD detector [39] and matched with those in the previous frame by the appearance-based descriptor LBD [40]. We use geometric constraints to remove outliers of line matches. For example, the distance between the midpoints of two matched lines should be no more than δ_{dist}^{th} pixels, and the angle difference should be no more than δ_{angle}^{th} degrees. After the feature detection and matching, the point features and the endpoints of line segments are projected onto the normalized image plane. Additionally, a frame is selected as a new keyframe if the average parallax of the tracked point features is larger than a threshold.

4.2. Back End

In the back-end thread, the points and lines are firstly triangulated to build re-projection residuals. In order to get good landmark estimations, the inverse depth of a point feature is estimated with all the observations. For line triangulation, we only choose two frames with the furthest spatial distance in the sliding window to initialize the Plücker coordinates.

After we get the initial estimation of map points/lines and the IMU body state predicted from IMU measurements, the sliding window optimization described in Section 3 is used to find the optimal states. To limit the size of the state vector $\mathcal{X}$, a two-way marginalization strategy is adopted to

remove states from the sliding window [23]. When the second newest frame $\mathbf{x}_{n+N-1}$ is a keyframe, we marginalize out the earliest frame x_n with all its measurements. Otherwise, if the second newest frame is not a keyframe, we discard the visual measurements from this frame and reserve its IMU measurements in the pre-integration measurements. New prior information is gained based on the marginalized measurements, reserving the constraint information from the removed states.

Lastly, we cull the invalid map points and lines. If the inverse depth of a point is negative, we will delete this point from the map. If the re-projection residuals of a line exceed a threshold it will be removed from the map.

4.3. Implementation Details

To bootstrap the VIO system, we adopt the visual–inertial alignment method [41] to recover the scale, gravity vector, initial biases, and velocity of the IMU initial body state. The sliding window is with $N = 10$ keyframes. Each frame has at most 150 point features, and 150 line segments. The thresholds used in line matching are set with $\delta_{dist}^{th} = 60$ pixels and $\delta_{angle}^{th} = 30°$. Since the visual–inertial system has only four unobservable DoFs (the yaw direction and the absolute position), the optimization methods for six DoFs may introduce illusory information into the roll and pitch directions by automatically taking steps along these directions to minimize the cost function. After the sliding window optimization, we reset the body state by rotating it back with the increments along the roll and pitch directions. All the numerical optimizations are solved using the Levenberg–Marquardt method in the `Ceres` solver library [42]. The line detection and matching codes are provided by OpenCV 3 [43].

5. Experimental Results

We evaluated our PL-VIO system using two public benchmark datasets: the EuRoc MAV Dataset [44] and the PennCOSYVIO Dataset [45].The accuracy of the PL-VIO method is compared with that of three state-of-the-art monocular VIO methods to validate the advantages of the PL-VIO method: ROVIO [17] and OKVIS [18] in monocular mode, and VINS-Mono [32] without loop closure. ROVIO is a tightly-coupled form of VIO based on the extended Kalman filter (EKF). It directly uses the intensity errors from images to find the optimal state during the update step. OKVIS is a sliding window optimization algorithm with point features which can work with monocular or stereo modes. VINS-Mono is a complete VIO-SLAM system employing point features to optimize IMU body states in a sliding window, and performs loop closure. All of the experiments were performed on the computer with an Intel Core i7-6700HQ CPU with 2.60 GHz, 16 GB RAM, and the ROS Kinetic [46].

5.1. EuRoc MAV Dataset

The EuRoc micro aerial vehicle (MAV) datasets were collected by an MAV in two indoor scenes, which contain stereo images from a global shutter camera at 20FPS and synchronized IMU measurements at 200 Hz [44]. Each dataset provides a ground truth trajectory given by the VICON motion capture system. All the extrinsic and intrinsic parameters are also provided in the datasets. In our experiments, we only used the images from the left camera.

The main advantage of line features is that they provide significant geometry structure information with respect to the environment. As an example, we show the reconstructed map built by PL-VIO from the MH_05_difficult sequence in Figure 5. The four images in Figure 5a–d were captured by an MAV flying in a machine hall, which showed the room's structure. As shown in Figure 5d, the line segment detection in the weak illumination scene was more robust than point feature detection. From the reconstructed 3D map, it can be seen that the geometry of the environment is described by the line segments, and thus semantic information could be extracted from the map. This is useful for robot navigation.
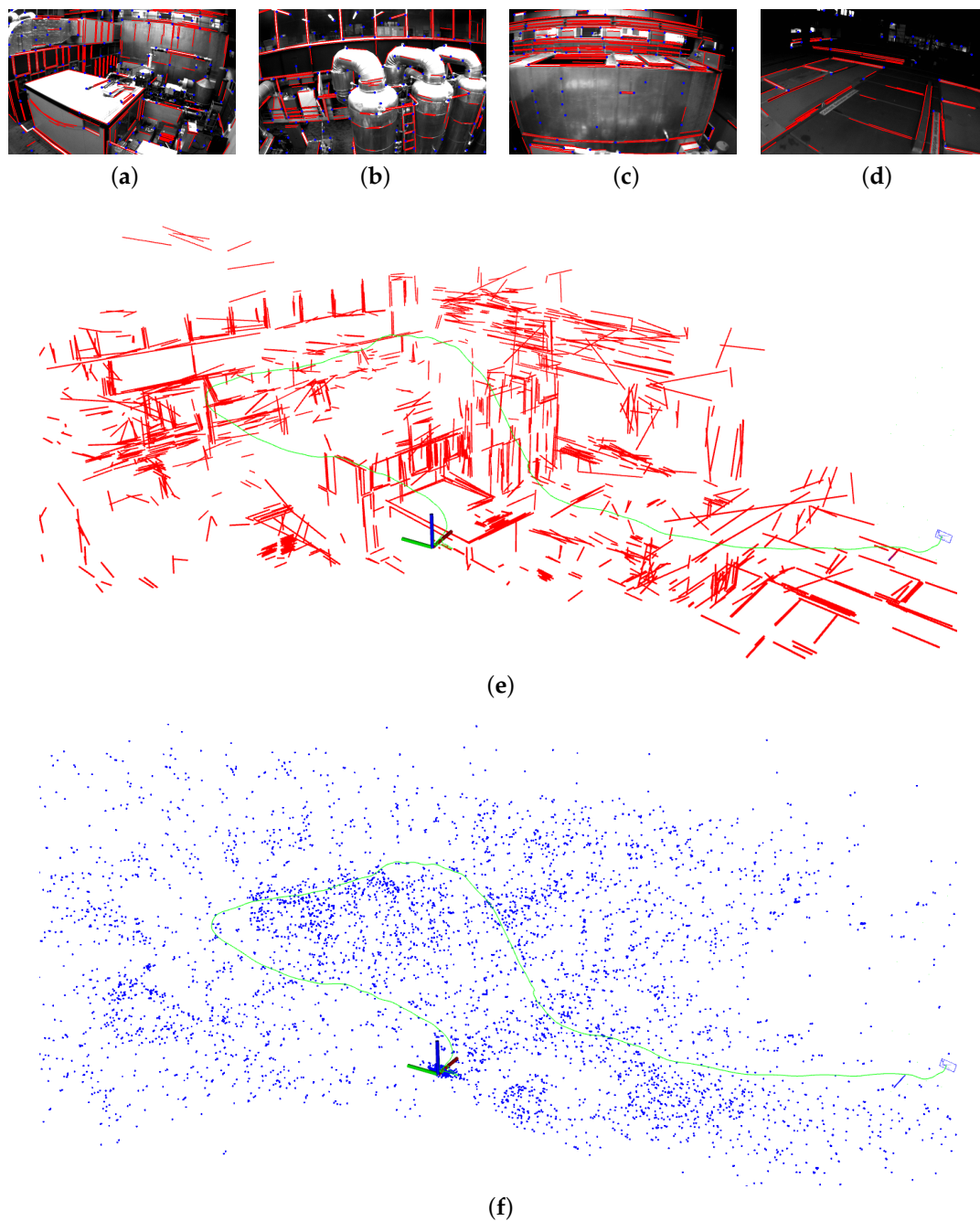
(**a**)　(**b**)　(**c**)　(**d**)

(**e**)

(**f**)

Figure 5. Mapping results in the MH_05_difficult sequence. (**a**–**d**) Detected point and line features for different frames. (**e**) The reconstructed line map (red) and the trajectory (green). The scenes with rich line features (such as the window, baluster, and cabinet) are clearly reconstructed from the map. (**f**) The reconstructed point map (blue). As compared to the line map, the structure is hard to identify in the point map.

For quantitative evaluation, we compared our PL-VIO with three state-of-the-art monocular visual–inertial methods: ROVIO [17], OKVIS [18] in monocular mode, and VINS-Mono [32] without loop closure. For the fair comparison, default parameters provided by the authors of these comparison methods were used. We chose the absolute pose error (APE) as the evaluation metric, which directly compared the trajectory error between the estimate and the ground truth [47]. The open-source package evo (https://michaelgrupp.github.io/evo/) provides an easy-to-use interface to evaluate the

trajectories of odometry and SLAM algorithms. We employed this tool to evaluate these methods in this section. Table 1 shows the root mean square error (RMSE) of translation and rotation along all the trajectory, and their histograms are also provided as shown in Figure 6.

Table 1. The root mean square error (RMSE) results on the several EuRoc MAV dataset. The translation (m) and rotation (rad) error are listed as follows. The numbers in bold represent the estimated trajectory is more close to the benchmark trajectory.

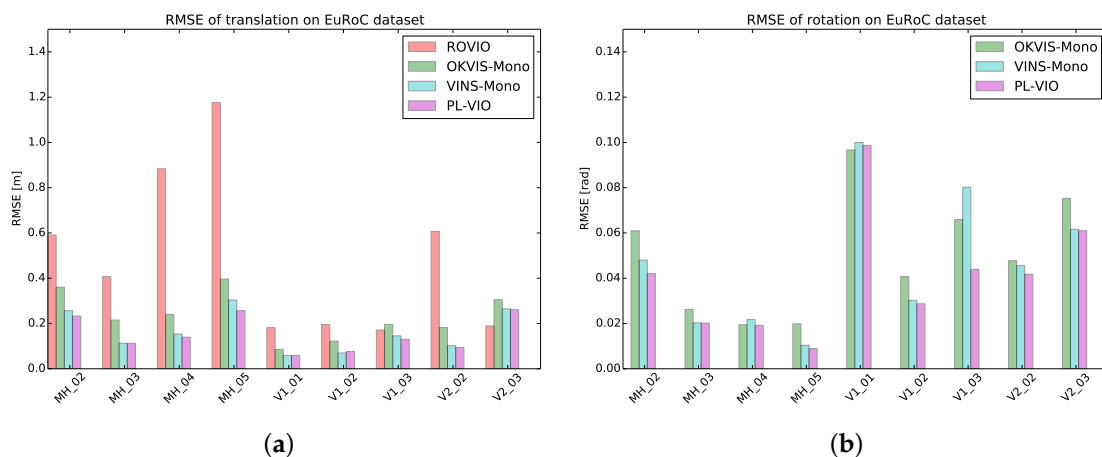| Seq.	ROVIO		OKVIS-Mono		VINS-Mono		PL-VIO	
	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.	Trans.	Rot.
MH_02_easy	0.59075	2.21181	0.36059	0.06095	0.25663	0.04802	**0.23274**	**0.04204**
MH_03_medium	0.40709	1.92561	0.21534	0.02622	0.11239	0.02027	**0.11224**	**0.02016**
MH_04_difficult	0.88363	2.30330	0.23984	0.01943	0.15366	0.02173	**0.13942**	**0.01915**
MH_05_difficult	1.17578	2.27213	0.39644	0.01987	0.30351	0.01038	**0.25687**	**0.00892**
V1_01_easy	0.18153	2.03399	0.08583	**0.09665**	**0.05843**	0.09995	0.05916	0.09869
V1_02_medium	0.19563	1.93652	0.12207	0.04073	**0.06970**	0.03022	0.07656	**0.02871**
V1_03_difficult	0.17091	2.02069	0.19613	0.06591	0.14531	0.08021	**0.13016**	**0.04382**
V2_02_medium	0.60686	1.84458	0.18253	0.04773	0.10218	0.04558	**0.09450**	**0.04177**
V2_03_difficult	**0.18912**	1.92380	0.30513	0.07527	0.26446	0.06162	0.26085	**0.06098**

Figure 6. RMSEs for ROVIO, OKVIS-Mono, and Vins-Mono without loop closure, and for the proposed PL-VIO using the `EuRoc` dataset. (**a**) RMSEs in translation. (**b**) RMSEs in rotation. The rotation errors of ROVIO are one order of magnitude higher than those of other three methods, so its result is not reported in (**b**).

Table 1 shows that our PL-VIO which jointly optimizes point and lines provided the best performance on eight sequences for the rotation, except for V1_01_easy. Our method also performed the best on six sequences for the translation, except for V1_01_easy, V1_02_medium, and V1_03_difficult. However, the difference with respect to the best results was only at the submillimeter level. The results in Table 1 show that integrating line features into VIO could improve the accuracy of motion estimation. To demonstrate the results intuitively, several heat map of trajectories estimated by PL-VIO and VINS-Mono are shown in Figure 7. The redder the color is, the larger the translation error is. Comparing the three trajectories, we came to the conclusion that PL-VIO with line features gave smaller errors than VINS-Mono when the camera was moved with rapid rotation. Furthermore, we found that these sequences with rapid rotation caused large changes in the viewing direction, and the lighting conditions are especially challenging for tracking point features [25,26,28]. As shown in Figure 8, 27 point pairs (including 10 outliers) were matched successfully, while 33 lines were matched successfully and all the matches are correct.
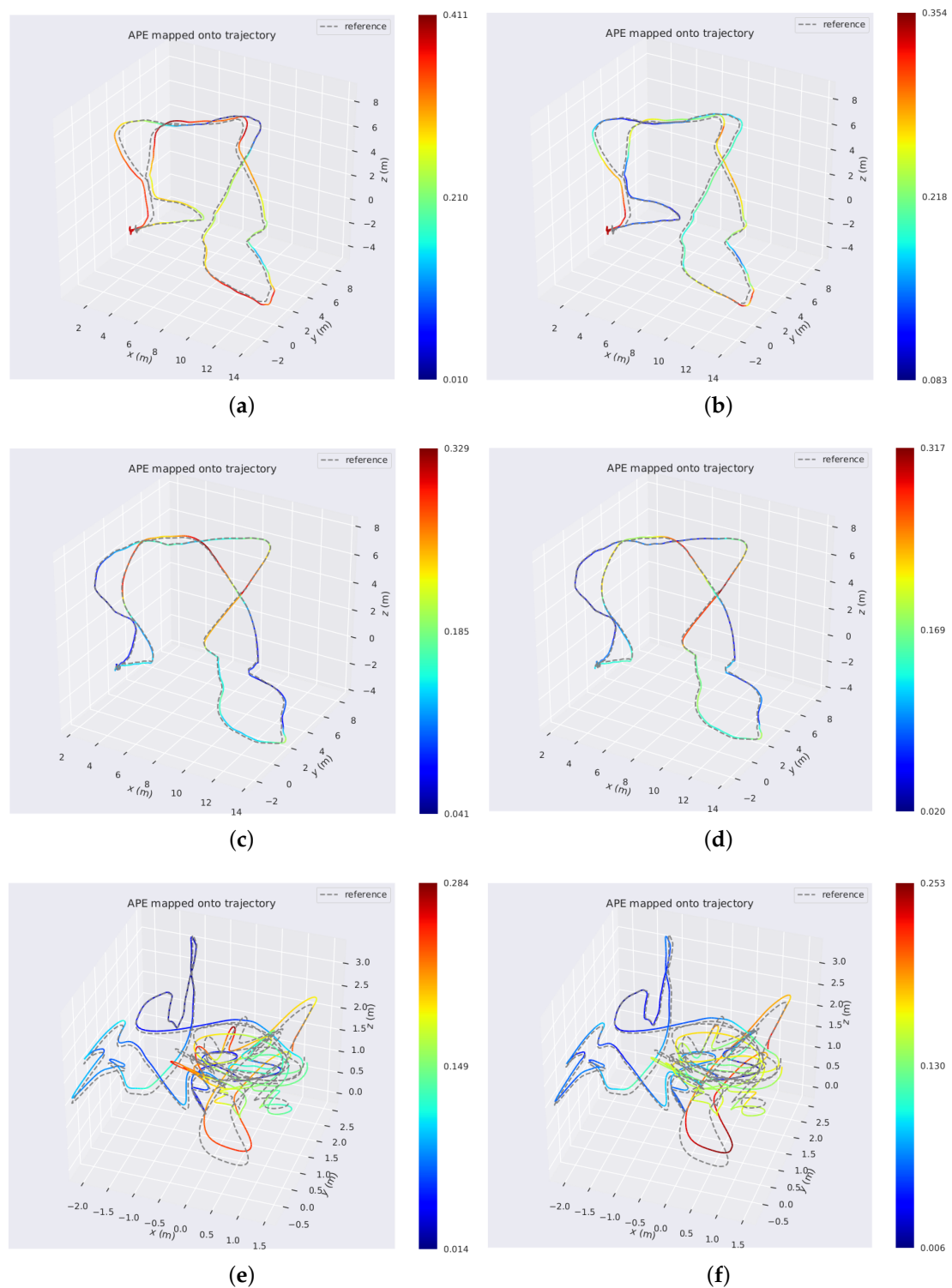
Figure 7. Comparison of the proposed method versus VINS-Mono. The three colorful trajectories of the left column are run with VINS-Mono on the (**a**) `MH_05_difficult`; (**c**) `MH_04_difficult`; and (**e**) `V1_03_difficult` sequences. The right three trajectories are the results of the proposed PL-VIO on the (**b**) `MH_05_difficult`; (**d**) `MH_04_difficult`; and (**f**) `V1_03_difficult` sequences. Colors encode the corresponding absolute pose errors.

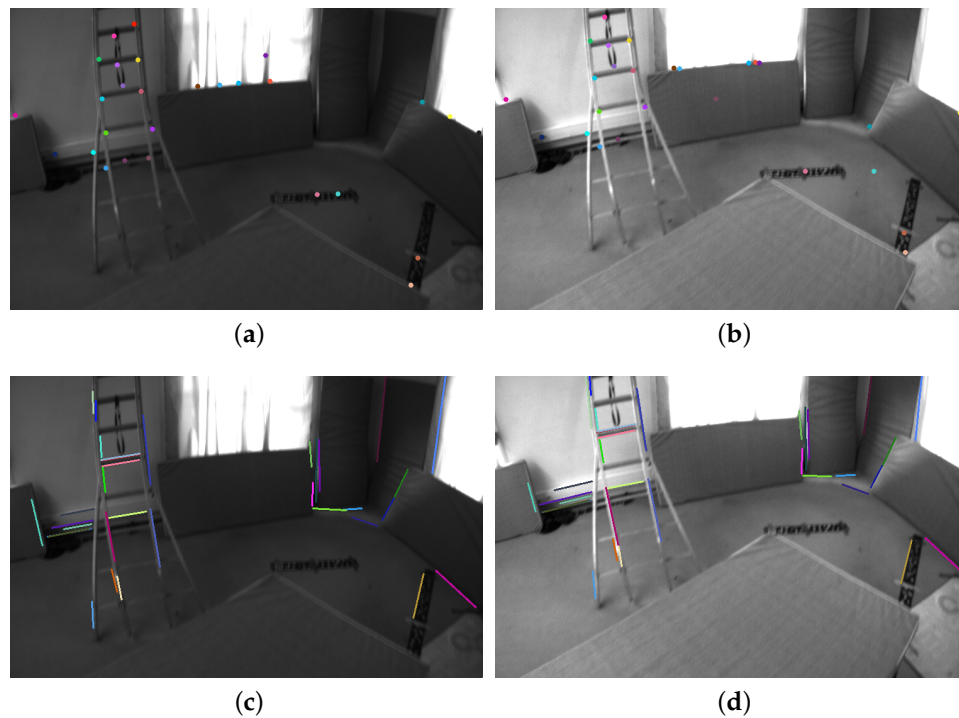(a)　　　　　　　　　　　　　　　　　　　　(b)

(c)　　　　　　　　　　　　　　　　　　　　(d)

Figure 8. Tracked point/line features with rapid rotation in `V1_03_difficult`. Lighting conditions and view directions changed in successive frames. Matched point features are marked with the same color in frame (**a**,**b**). Matched line segment features are marked with the same color in frame (**c**,**d**).

Besides, the computation times of different methods are listed in Table 2. The computation efficiency of filter-based ROVIO was the highest, while its accuracy was the lowest. The proposed PL-VIO system was the most time-consuming method, but its computation time was mainly restricted by the line detection and matching step. In Section 5.3, the computation times of different modules in PL-VIO are independently estimated in the V1_03_difficult sequence, and it was found that the computation efficiency of the PL-VIO system directly depended on the line detection and matching.

Table 2. Average running times of different methods on the EuRoc dataset (unit: millisecond).

Seq.	ROVIO	OKVIS-Mono	VINS-Mono	PL-VIO
MH_02_easy	15	31	63	127
MH_03_medium	15	30	62	112
MH_04_difficult	15	24	54	108
MH_05_difficult	15	27	58	102
V1_01_easy	14	26	45	93
V1_02_medium	15	23	37	86
V1_03_difficult	15	20	29	82
V2_02_medium	15	22	33	86
V2_03_difficult	15	21	27	85

5.2. PennCOSYVIO Dataset

The PennCOSYVIO dataset is a recent VIO benchmark that collects the synchronized data of a large glass building with hand-held equipment from outdoors to indoors (see Figure 9) [45]. Challenging factors include illumination changes, rapid rotations, and repetitive structures. All the intrinsic and extrinsic parameters as well as the ground truth trajectory are provided. We use data collected by the VI-sensor, which was also used in the EuRoc MAV datasets.
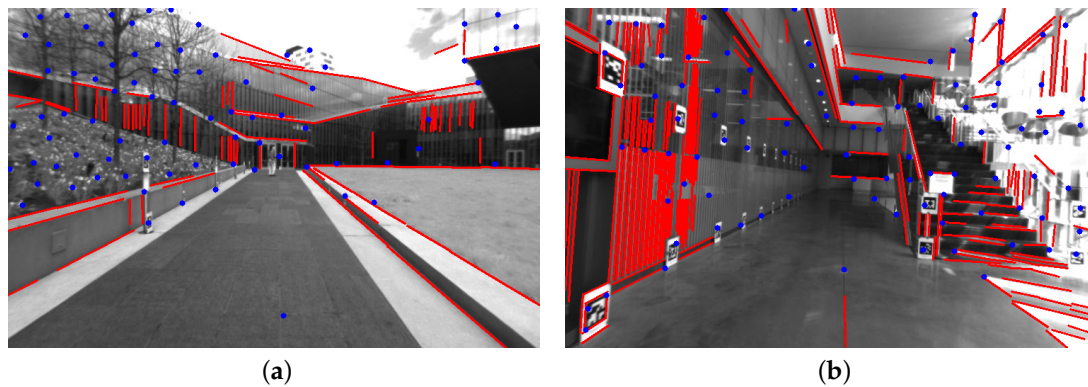
(a)   **(b)**

Figure 9. Images from PennCOSYVIO dataset. Red lines are detected lines. (**a**) Outside the glass building. (**b**) Inside the glass building.

We compared the proposed PL-VIO with the VINS-Mono without loop closure. To evaluate fairly, the same parameters were used for PL-VIO and VINS-Mono. The same metric and evaluation tools used in Section 5.1 were employed here to evaluate the trajectories. Table 3 lists the results.

Table 3. The RMSE of the absolute pose error (APE) for different algorithms. The numbers in bold represent the estimated trajectory is more close to the benchmark trajectory.

Algorithm	Translation Error (m)	Rotation Error (rad)
VINS-Mono	1.14690	0.04156
PL-VIO	**1.05975**	**0.03742**

Furthermore, the evaluation tool (https://daniilidis-group.github.io/penncosyvio/) is also provided in the PennCOSYVIO dataset, and adopts two metrics, the APE and relative pose error (RPE). For RPE, it expresses the errors in percentages by dividing the value with the path length [45]. The creators of PennCOSYVIO cautiously selected the evaluation parameters, so their tool is suited for evaluating VIO approaches in this dataset. Therefore, we adopted this evaluation tool in our experiments, and the results are listed in Table 4.

Table 4. The results evaluated by PennCOSYVIO evaluation tool for different algorithms. The rotation errors for the APE and relative pose error (RPE) are expressed in degrees. The translation error is expressed in the x-axis, y-axis, and z-axis. The APE is expressed in meters, while the RPE is expressed in percentages (%). The numbers in bold represent the estimated trajectory is more close to the benchmark trajectory.

Algorithm	APE				RPE			
	x	y	z	Rot.	x	y	z	Rot.
VINS-Mono	**0.423**	0.173	0.861	2.3477	2.807	1.844	4.663	1.9337
PL-VIO	0.524	**0.070**	**0.769**	**2.0782**	**2.375**	1.844	**4.361**	**1.7350**

From Tables 3 and 4, it can be seen that the PL-VIO obtained the best performance for the rotation part. The APE of translation evaluated by PennCOSYVIO tool provided more details. Compared to VINS-Mono, PL-VIO gave smaller errors in the y-axis and z-axis, and a smaller error summation of the three axes. VINS-Mono obtained better performance only in the x-axis.

5.3. Computing Time

Finally, we evaluated the average execution time of our PL-VIO running at the `V1_02_medium` sequence because this image sequence was collected from a typical indoor scene. Table 5 shows the execution time of each block. We can see that line detection and matching, which runs at 11 Hz in the front end, is the bottleneck in terms of efficiency. State-of-the-art line detection and matching methods, such as the combination of LSD and LBD, are not satisfactory for VIO/SLAM systems. Note that our method is independent of line feature detection and matching, so improving their efficiency is beyond the scope of this paper. Marginalization in the back end is another time-consuming part. We observe that the inefficiency of marginalization is caused by the fill-in when marginalizing out features, which makes the Hessian matrix become a less sparse matrix. This problem can be potentially solved by discarding some features when performing marginalization to maintain a sparse Hessian matrix [18].

Table 5. Mean execution time of PL-VIO run with the `V1_02_medium` sequence.

Module	Operation	Times (ms)	Rate (Hz)	Thread ID
front end	point feature detection and matching	4	25	1
	line feature detection and matching	86	11	2
	IMU forward propagation	1	100	3
back end	nonlinear optimization	28	15	4
	marginalization	35	15	4
	feature triangulation and culling	2	15	4

6. Conclusions

This paper presents the novel tightly-coupled monocular vision-inertial odometry algorithm PL-VIO, which optimizes the system states in a sliding window with both point and line features. The proposed PL-VIO system has two main modules: the front end and the back end. The front-end module is used to propagate IMU body state, and detect and match point/line features. The back-end module is used to estimate and optimize the body states. In the back-end module, a line landmark is considered as an infinite 3D spatial line and its orthonormal representation is employed to parameterize it compactly during optimization. Furthermore, all the Jacobian matrices of error terms are given in detail for solving the sliding window optimization efficiently. We also provide the evaluation results of the proposed PL-VIO as compared to three state-of-the-art monocular VIO methods including ROVIO [17], OKVIS [18], and VINS-Mono [32] on both the `EuRoc` dataset and PennCOSYVIO dataset. According to the analysis and results, two further conclusions are as follows:

1. The reconstructed 3D map with line features can provide geometrical information with respect to the environment, and thus semantic information could be extracted from the map. This is useful for robot navigation.

2. Line features can improve the system accuracy both for translation and rotation, especially in illumination-changing scenes. However, the line detection and matching are time-consuming and become the bottlenecks in the efficiency of the system.

In the future, we plan to improve our system by introducing the structural constraints between 3D spatial lines, such as parallel or coplanar lines in Manhattan-world scenes [48]. Geometric constraints among these lines have the potential to further improve localization precision and reduce rotation accumulation errors.

Author Contributions: Yijia He and Ji Zhao conceived and designed the alogrithm; Yijia He performed the experiments, analyzed the data, and drafted the paper; Yue Guo and Wenhao He contributed analysis tools; Ji Zhao and Kui Yuan revised the manuscript.

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

Appendix A

The IMU error state propagtion equation can be defined as [49]:

$$
\begin{bmatrix}
\delta\boldsymbol{\alpha}_{b_{k+1}b'_{k+1}} \\
\delta\boldsymbol{\theta}_{b_{k+1}b'_{k+1}} \\
\delta\boldsymbol{\beta}_{b_{k+1}b'_{k+1}} \\
\delta\mathbf{b}_a^{b_{k+1}} \\
\delta\mathbf{b}_g^{b_{k+1}}
\end{bmatrix}
= \mathcal{F}_k
\begin{bmatrix}
\delta\boldsymbol{\alpha}_{b_k b'_k} \\
\delta\boldsymbol{\theta}_{b_k b'_k} \\
\delta\boldsymbol{\beta}_{b_k b'_k} \\
\delta\mathbf{b}_a^{b_k} \\
\delta\mathbf{b}_g^{b_k}
\end{bmatrix}
+ \mathcal{G}_k
\begin{bmatrix}
\mathbf{n}_a^{b_k} \\
\mathbf{n}_g^{b_k} \\
\mathbf{n}_a^{b_{k+1}} \\
\mathbf{n}_g^{b_{k+1}} \\
\mathbf{n}_{\mathbf{b}_a} \\
\mathbf{n}_{\mathbf{b}_g}
\end{bmatrix}
\tag{A1}
$$

With

$$
\mathcal{F}_k =
\begin{bmatrix}
\mathbf{I} & f_{12} & \mathbf{I}\delta t & -\frac{1}{4}(\mathbf{q}_{b_i b_k} + \mathbf{q}_{b_i b_{k+1}})\delta t^2 & f_{15} \\
0 & I - [\boldsymbol{\omega}]_\times & 0 & 0 & -\mathbf{I}\delta t \\
0 & f_{32} & \mathbf{I} & -\frac{1}{2}(\mathbf{q}_{b_i b_k} + \mathbf{q}_{b_i b_{k+1}})\delta t & f_{35} \\
0 & 0 & 0 & \mathbf{I} & 0 \\
0 & 0 & 0 & 0 & \mathbf{I}
\end{bmatrix}
\tag{A2}
$$

$$
\mathcal{G}_k =
\begin{bmatrix}
\frac{1}{4}\mathbf{q}_{b_i b_k}\delta t^2 & g_{12} & \frac{1}{4}\mathbf{q}_{b_i b_{k+1}}\delta t^2 & g_{14} & 0 & 0 \\
0 & \frac{1}{2}\delta t & 0 & \frac{1}{2}\delta t & 0 & 0 \\
\frac{1}{2}\mathbf{q}_{b_i b_k}\delta t & g_{32} & \frac{1}{2}\mathbf{q}_{b_i b_{k+1}}\delta t & g_{34} & 0 & 0 \\
0 & 0 & 0 & 0 & \delta t & 0 \\
0 & 0 & 0 & 0 & 0 & \delta t
\end{bmatrix}
\tag{A3}
$$

$$
f_{12} = \frac{\partial\boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial\delta\boldsymbol{\theta}_{b_k b'_k}} = -\frac{1}{4}(\mathbf{R}_{b_i b_k}[\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k}]_\times\delta t^2 + \mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times(\mathbf{I} - [\boldsymbol{\omega}]_\times\delta t)\delta t^2)
$$

$$
f_{32} = \frac{\partial\boldsymbol{\beta}_{b_i b_{k+1}}}{\partial\delta\boldsymbol{\theta}_{b_k b'_k}} = -\frac{1}{2}(\mathbf{R}_{b_i b_k}[\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k}]_\times\delta t + \mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times(\mathbf{I} - [\boldsymbol{\omega}]_\times\delta t)\delta t)
$$

$$
f_{15} = \frac{\partial\boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial\delta\mathbf{b}_g^{b_k}} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times\delta t^2)(-\delta t)
$$

$$
\tag{A4}
$$

$$
f_{35} = \frac{\partial\boldsymbol{\beta}_{b_i b_{k+1}}}{\partial\delta\mathbf{b}_g^{b_k}} = -\frac{1}{2}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times\delta t)(-\delta t)
$$

$$
g_{12} = \frac{\partial\boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial\mathbf{n}_g^{b_k}} = g_{14} = \frac{\partial\boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial\mathbf{n}_g^{b_{k+1}}} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times\delta t^2)(\frac{1}{2}\delta t)
$$

$$
g_{32} = \frac{\partial\boldsymbol{\beta}_{b_i b_{k+1}}}{\partial\mathbf{n}_g^{b_k}} = g_{34} = \frac{\partial\boldsymbol{\beta}_{b_i b_{k+1}}}{\partial\mathbf{n}_g^{b_{k+1}}} = -\frac{1}{2}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_a^{b_k})]_\times\delta t^2)(\frac{1}{2}\delta t)
$$

We can define the error state vector with $\boldsymbol{\eta}_{k+1} = [\delta\boldsymbol{\alpha}_{b_{k+1}b'_{k+1}}, \delta\boldsymbol{\theta}_{b_{k+1}b'_{k+1}}, \delta\boldsymbol{\beta}_{b_{k+1}b'_{k+1}}, \delta\mathbf{b}_a^{b_{k+1}}, \delta\mathbf{b}_g^{b_{k+1}}]^\top$. The noise vector is $\mathbf{n}_k = [\mathbf{n}_a^{b_k}, \mathbf{n}_g^{b_k}, \mathbf{n}_a^{b_{k+1}}, \mathbf{n}_g^{b_{k+1}}, \mathbf{n}_{\mathbf{b}_a}, \mathbf{n}_{\mathbf{b}_g}]^\top$. Equation (A1) can be written in compact matrix form as:

$$
\boldsymbol{\eta}_{k+1} = \mathcal{F}_k\boldsymbol{\eta}_k + \mathcal{G}_k\mathbf{n}_k
\tag{A5}
$$

The pre-integrated measurements covariance can be computed iteratively based on the linear model (A5) [14]:

$$
\Sigma_{b_i b_{k+1}} = \mathcal{F}_k\Sigma_{b_i b_k}\mathcal{F}_k^\top + \mathcal{G}_k\Sigma_\mathbf{n}\mathcal{G}_k^\top
\tag{A6}
$$

where $\Sigma_{\mathbf{n}}$ is the covariance of the raw IMU measurements, and the initial covariance is $\Sigma_{b_i b_i} = \mathbf{0}_{15 \times 15}$. Also we can compute the Jacobian matrix of pre-integrated measurements $\mathbf{z}_{b_i b_j}$ with respect to error state $\boldsymbol{\eta}_i$ iteratively with [49]:

$$\mathbf{J}_{ik+1} = \mathcal{F}_k \mathbf{J}_{ik} \tag{A7}$$

With $\mathbf{J}_{ii} = \mathbf{I}$. These Jacobian matrices given in Equation (10) can be extracted from $\mathbf{J}_{ij}$.

Appendix B

The Jacobian matrix of the line re-projection error respect to the orthonormal representation is:

$$
\begin{aligned}
\frac{\partial \mathbf{r}_l}{\partial \mathbf{l}} &= \begin{bmatrix} \frac{\partial r_1}{\partial l_1} & \frac{\partial r_1}{\partial l_2} & \frac{\partial r_1}{\partial l_3} \\ \frac{\partial r_2}{\partial l_1} & \frac{\partial r_2}{\partial l_2} & \frac{\partial r_2}{\partial l_3} \end{bmatrix} \\
&= \begin{bmatrix} \frac{-l_1 \mathbf{s}_l^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{u_s}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{-l_2 \mathbf{s}_l^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{v_s}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{1}{(l_1^2+l_2^2)^{(\frac{1}{2})}} \\ \frac{-l_1 \mathbf{e}_l^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{u_e}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{-l_2 \mathbf{e}_l^\top \mathbf{1}}{(l_1^2+l_2^2)^{(\frac{3}{2})}} + \frac{v_e}{(l_1^2+l_2^2)^{(\frac{1}{2})}} & \frac{1}{(l_1^2+l_2^2)^{(\frac{1}{2})}} \end{bmatrix}_{2 \times 3}
\end{aligned} \tag{A8}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}_w}{\partial \delta \mathcal{O}} &= \begin{bmatrix} \frac{\partial \mathcal{L}_w}{\partial \mathbf{u}_1} & \frac{\partial \mathcal{L}_w}{\partial \mathbf{u}_2} & \frac{\partial \mathcal{L}_w}{\partial w_1} & \frac{\partial \mathcal{L}_w}{\partial w_2} \end{bmatrix}_{6 \times 8} \begin{bmatrix} \frac{\partial \mathbf{u}_1}{\partial \delta \boldsymbol{\psi}} & \frac{\partial \mathbf{u}_1}{\partial \delta \phi} \\ \frac{\partial \mathbf{u}_2}{\partial \delta \boldsymbol{\psi}} & \frac{\partial \mathbf{u}_2}{\partial \delta \phi} \\ \frac{\partial w_1}{\partial \delta \boldsymbol{\psi}} & \frac{\partial w_1}{\partial \delta \phi} \\ \frac{\partial w_2}{\partial \delta \boldsymbol{\psi}} & \frac{\partial w_2}{\partial \delta \phi} \end{bmatrix}_{8 \times 4} \\
&= \begin{bmatrix} w_1 \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{u}_1 & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{3 \times 3} & w_2 \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 1} & \mathbf{u}_2 \end{bmatrix}_{6 \times 8} \begin{bmatrix} \mathbf{0} & -\mathbf{u}_3 & \mathbf{u}_2 & \mathbf{0} \\ \mathbf{u}_3 & \mathbf{0} & -\mathbf{u}_1 & \mathbf{0} \\ 0 & 0 & 0 & -w_2 \\ 0 & 0 & 0 & w_1 \end{bmatrix}_{8 \times 4} \\
&= \begin{bmatrix} \mathbf{0} & -w_1 \mathbf{u}_3 & w_1 \mathbf{u}_2 & -w_2 \mathbf{u}_1 \\ w_2 \mathbf{u}_3 & \mathbf{0} & -w_2 \mathbf{u}_1 & w_1 \mathbf{u}_2 \end{bmatrix}_{6 \times 4}
\end{aligned} \tag{A9}
$$

To compute the line re-projection error, a spatial line in world frame w is transformed to the body frame b firstly, and then transformed to the camera frame c with the extrinsic parameters $\mathbf{T}_{bc}$.

$$
\begin{aligned}
\mathcal{L}_c &= \mathcal{T}_{bc}^{-1} \mathcal{T}_{wb}^{-1} \mathcal{L}_w \\
&= \mathcal{T}_{bc}^{-1} \begin{bmatrix} \mathbf{R}_{wb}^\top (\mathbf{n}^w + [\mathbf{d}^w]_\times \mathbf{p}_{wb}) \\ \mathbf{R}_{wb}^\top \mathbf{d}^w \end{bmatrix}_{6 \times 1}
\end{aligned} \tag{A10}
$$

The Jacobian matrix of the line re-projection error with respect to rotation of the ith IMU body state is:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_c}{\partial \delta \boldsymbol{\theta}_{bb'}} &= \mathcal{T}_{bc}^{-1} \begin{bmatrix} \frac{\partial (\mathbf{I} - [\delta \boldsymbol{\theta}_{bb'}]_\times) \mathbf{R}_{wb}^\top (\mathbf{n}^w + [\mathbf{d}^w]_\times \mathbf{p}_{wb})}{\partial \delta \boldsymbol{\theta}_{bb'}} \\ \frac{\partial (\mathbf{I} - [\delta \boldsymbol{\theta}_{bb'}]_\times) \mathbf{R}_{wb}^\top \mathbf{d}^w}{\partial \delta \boldsymbol{\theta}_{bb'}} \end{bmatrix} \\
&= \mathcal{T}_{bc}^{-1} \begin{bmatrix} [\mathbf{R}_{wb}^\top (\mathbf{n}^w + [\mathbf{d}^w]_\times \mathbf{p}_{wb})]_\times \\ [\mathbf{R}_{wb}^\top \mathbf{d}^w]_\times \end{bmatrix}_{6 \times 3}
\end{aligned} \tag{A11}
$$

The Jacobian matrix of the line re-projection error with respect to position of the *i*th IMU body state is as follows:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_c}{\partial \delta \mathbf{p}_{bb'}} &= \mathcal{T}_{bc}^{-1}
\begin{bmatrix}
\frac{\partial \mathbf{R}_{wb}^{\top}(\mathbf{n}^w + [\mathbf{d}^w]_{\times}(\mathbf{p}_{wb} + \delta \mathbf{p}_{bb'}))}{\partial \delta \mathbf{p}_{bb'}} \\
\frac{\partial \mathbf{R}_{wb}^{\top}\mathbf{d}^w}{\partial \delta \mathbf{p}_{bb'}}
\end{bmatrix} \\
&= \mathcal{T}_{bc}^{-1}
\begin{bmatrix}
\mathbf{R}_{wb}^{\top}[\mathbf{d}^w]_{\times} \\
\mathbf{0}
\end{bmatrix}_{6 \times 3}
\end{aligned}
\tag{A12}
$$

References

1. Groves, P.D. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*; Artech House: Norwood, MA, USA, 2013.
2. Martínez, J.L.; Morán, M.; Morales, J.; Reina, A.J.; Zafra, M. Field Navigation Using Fuzzy Elevation Maps Built with Local 3D Laser Scans. *Appl. Sci.* **2018**, *8*, 397.
3. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
4. Liu, L.; Mei, T.; Niu, R.; Wang, J.; Liu, Y.; Chu, S. RBF-Based Monocular Vision Navigation for Small Vehicles in Narrow Space below Maize Canopy. *Appl. Sci.* **2016**, *6*, 182.
5. Valiente, D.; Gil, A.; Payá, L.; Sebastián, J.M.; Reinoso, Ó. Robust Visual Localization with Dynamic Uncertainty Management in Omnidirectional SLAM. *Appl. Sci.* **2017**, *7*, 1294.
6. Borraz, R.; Navarro, P.J.; Fernández, C.; Alcover, P.M. Cloud Incubator Car: A Reliable Platform for Autonomous Driving. *Appl. Sci.* **2018**, *8*, 303.
7. Wang, X.; Wang, J. Detecting glass in simultaneous localisation and mapping. *Robot. Auton. Syst.* **2017**, *88*, 97–103.
8. Titterton, D.; Weston, J.L. *Strapdown Inertial Navigation Technology*; The Institution of Engineering and Technology: Stevenage, UK, 2004; Volume 17.
9. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332.
10. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262.
11. Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625.
12. Gui, J.; Gu, D.; Wang, S.; Hu, H. A review of visual inertial odometry from filtering and optimisation perspectives. *Adv. Robot.* **2015**, *29*, 1289–1301.
13. Liu, Y.; Chen, Z.; Zheng, W.; Wang, H.; Liu, J. Monocular Visual-Inertial SLAM: Continuous Preintegration and Reliable Initialization. *Sensors* **2017**, *17*, 2613.
14. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21.
15. Weiss, S.; Siegwart, R. Real-time metric state estimation for modular vision-inertial systems. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 4531–4537.
16. Kneip, L.; Weiss, S.; Siegwart, R. Deterministic initialization of metric state estimation filters for loosely-coupled monocular vision-inertial systems. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, 25–30 September 2011; pp. 2235–2241.
17. Bloesch, M.; Burri, M.; Omari, S.; Hutter, M.; Siegwart, R. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *Int. J. Robot. Res.* **2017**, *36*, 1053–1072.
18. 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.
19. 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.

20. Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R. Robust visual inertial odometry using a direct EKF-based approach. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 298–304.
21. Mourikis, A.I.; Roumeliotis, S.I. A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3565–3572.
22. 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.
23. Shen, S.; Michael, N.; Kumar, V. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 5303–5310.
24. Mur-Artal, R.; Tardós, J.D. Visual-inertial monocular SLAM with map reuse. *IEEE Robot. Autom. Lett.* **2017**, *2*, 796–803.
25. Kong, X.; Wu, W.; Zhang, L.; Wang, Y. Tightly-coupled stereo visual-inertial navigation using point and line features. *Sensors* **2015**, *15*, 12816–12833.
26. Kottas, D.G.; Roumeliotis, S.I. Efficient and consistent vision-aided inertial navigation using line observations. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 1540–1547.
27. Zhang, G.; Lee, J.H.; Lim, J.; Suh, I.H. Building a 3-D Line-Based Map Using Stereo SLAM. *IEEE Trans. Robot.* **2015**, *31*, 1364–1377.
28. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508.
29. Gomez-Ojeda, R.; Moreno, F.A.; Scaramuzza, D.; Gonzalez-Jimenez, J. PL-SLAM: A Stereo SLAM System through the Combination of Points and Line Segments. *arXiv* **2017**, arXiv:1705.09479.
30. Bartoli, A.; Sturm, P. The 3D line motion matrix and alignment of line reconstructions. *Int. J. Comput. Vis.* **2004**, *57*, 159–178.
31. Zuo, X.; Xie, X.; Liu, Y.; Huang, G. Robust Visual SLAM with Point and Line Features. *arXiv* **2017**, arXiv:1711.08654.
32. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *arXiv* **2017**, arXiv:1708.03852.
33. Furgale, P.; Rehder, J.; Siegwart, R. Unified temporal and spatial calibration for multi-sensor systems. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 1280–1286.
34. Kok, M.; Hol, J.D.; Schön, T.B. Using inertial sensors for position and orientation estimation. *arXiv* **2017**, arXiv:1704.06053.
35. Kaess, M.; Johannsson, H.; Roberts, R.; Ila, V.; Leonard, J.J.; Dellaert, F. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Int. J. Robot. Res.* **2012**, *31*, 216–235.
36. Sola, J. Quaternion kinematics for the error-state Kalman filter. *arXiv* **2017**, arXiv:1711.02508.
37. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI), Vancouver, BC, Canada, 24–28 August 1981.
38. Rosten, E.; Porter, R.; Drummond, T. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 105–119.
39. Von Gioi, R.G.; Jakubowicz, J.; Morel, J.M.; Randall, G. LSD: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 722–732.
40. Zhang, L.; Koch, R. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *J. Vis. Commun. Image Represent.* **2013**, *24*, 794–805.
41. Yang, Z.; Shen, S. Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 39–51.
42. Agarwal, S.; Mierle, K. Ceres Solver. Available online: http://ceres-solver.org (accessed on 9 April 2018).
43. Kaehler, A.; Bradski, G. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2016.

44. 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.

45. Pfrommer, B.; Sanket, N.; Daniilidis, K.; Cleveland, J. PennCOSYVIO: A challenging visual inertial odometry benchmark. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June2017; pp. 3847–3854.

46. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. *ROS: An Open-Source Robot Operating System*; ICRA Workshop on Open Source Software; ICRA: Kobe, Japan, 2009; p. 5.

47. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal , 7–12 October 2012; pp. 573–580.

48. Lu, Y.; Song, D.; Yi, J. High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 1540–1545.

49. Yang, Z.; Shen, S. *Tightly-Coupled Visual–Inertial Sensor Fusion Based on IMU Pre-Integration*; Technical Report; Hong Kong University of Science and Technology: Hong Kong, China, 2016.