applied sciences

Article

Semi-Direct Point-Line Visual Inertial Odometry for MAVs

Bo Gao, Baowang Lian and Chengkai Tang *

School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China
* Correspondence: cktang@nwpu.edu.cn

Abstract: Traditional Micro-Aerial Vehicles (MAVs) are usually equipped with a low-cost Inertial Measurement Unit (IMU) and monocular cameras, how to achieve high precision and high reliability navigation under the framework of low computational complexity is the main problem for MAVs. To this end, a novel semi-direct point-line visual inertial odometry (SDPL-VIO) has been proposed for MAVs. In the front-end, point and line features are introduced to enhance image constraints and increase environmental adaptability. At the same time, the semi-direct method combined with IMU pre-integration is used to complete motion estimation. This hybrid strategy combines the accuracy and loop closure detection performance of the feature-based method with the rapidity of the direct method, and tracks keyframes and non-keyframes, respectively. In the back-end, the sliding window mechanism is adopted to limit the computation, while the improved marginalization method is used to decompose the high-dimensional matrix corresponding to the cost function to reduce the computational complexity in the optimization process. The comparison results in the EuRoC datasets demonstrate that SDPL-VIO performs better than the other state-of-the-art visual inertial odometry (VIO) methods, especially in terms of accuracy and real-time performance.

Keywords: point-line feature; semi-direct; tracking; visual inertial odometry; marginalization

check for updates

Citation: Gao, B.; Lian, B.; Tang, C. Semi-Direct Point-Line Visual Inertial Odometry for MAVs. *Appl. Sci.* **2022**, *12*, 9265. https://doi.org/10.3390/app12189265

Academic Editors: Marcin Uradzinski, You Li and Hang Guo

Received: 22 August 2022
Accepted: 13 September 2022
Published: 15 September 2022

1. Introduction

A navigation system is one of the main applications for MAVs [1], mainly providing accurate and reliable attitude, speed and position information, and is indispensable in the process of MAVs flight and control. Due to limitations in size and payload capacity, MAVs are typically equipped with low-cost sensors and lack the hardware required to run advanced integrated navigation algorithms. Designing a low-cost integrated navigation platform for MAVs and building a more efficient system on this basis to ensure accuracy and real-time performance is the basic premise for MAVs to perform tasks and even survive.

Visual Inertial Navigation (VIN) is a hot topic in the field of MAVs research [2,3]. Cameras can provide abundant visual information, and inertial sensors (gyroscopes and accelerometers) can provide short-term and high-precision pose estimation. Their low cost, low power, small size, and complementarity make a combination of them particularly suitable for MAVs.

Based on the functional structure, VINs can be classified into the front end and the back end. The front end completes the calculation of visual and inertial motion states, and the back end realizes data fusion and outputs the optimal state estimation.

Front-end image processing methods can be divided into the following: (1) Feature-based methods [4–12], which extract representative features (such as point or line features) from images, and then match them according to the description of features. OKVIS [5] finds features using the Harris corner detector, and matches them using Binary robust invariant scalable keypoints (BRISK) descriptors. ORB-SLAM2 [7] performs feature extraction, description and matching, and then performs motion estimation using Oriented FAST and Rotated BRIEF (ORB) features. PL-SLAM [8] integrates the line representation within the SLAM, and improves the performance of ORB-SLAM2, especially in poorly textured environments. However, feature extraction and the calculation of descriptors are very time-consuming. (2) Direct methods [13,14], which estimate the motion based on the pixel gray difference between two images. DSO [14] minimizes the photometric error to estimate the camera motion, which greatly reduces the amount of computation compared with feature-based methods. However, it has high requirements regarding image quality and is not suitable for large inter-frame motion. (3) Semi-direct methods [15–22], which combine the above two methods and have received increasing attention from researchers in recent years. SVO [15] first uses the image intensity to estimate the pose, and then uses the position of feature points to optimize the pose. However, there are still shortcomings in motion-tracking accuracy and robustness. PL-SVO [16] extends line segments to the SVO algorithm, and has stronger robustness in poorly textured environments. SVL [18] combines ORB-SLAM and SVO, in which the former is used in keyframes and the latter is used in non-keyframes. PCSD-VIO [22] refers to the frame-tracking strategy of SVL, but integrates online photometric calibration, and the fusion method with IMU is slightly different.

Back-end data fusion methods can be divided into the following: (1) Filtering-based methods [4,6], which use inertial observation for state propagation and visual observation for state update. As the number of features in the state vector increases, the computational complexity rapidly increases, so it is not suitable for a large range of scenes. (2) Optimization-based methods [5,9–11,23–28], which are usually based on keyframes and minimize the overall errors by establishing the connection relation between frames and constantly adjusting the pose of frames. VINS-Mono [10] is a tightly coupled algorithm based on nonlinear optimization, with initialization, loop detection and relocalization modules. PL-VIO [11] adds line features to the basic frame of VINS-Mono, and has achieved good results. However, it lacks a loop-detection module, and uses a large amount of computation in nonlinear optimization.

The above work has both advantages and disadvantages, such as the adoption of time-consuming feature extraction, sensitivity to poorly textured environments and large illumination changes, ability to work without inertial sensors, and the large amount of computation needed in nonlinear optimization. Therefore, a novel, semi-direct, point-line, visual inertial odometry for MAVs has been proposed. The main contributions are as follows:

Firstly, motion estimation is accomplished using the semi-direct method, which realizes the mutual advantage compensation of the feature-based method and the direct method, that is, the former accurately tracks keyframes, and extracts point and line features for back-end nonlinear optimization and loop-closure detection, while the latter rapidly tracks non-keyframes through direct image alignment.

Secondly, the sliding window mechanism is adopted to effectively combine visual and inertial information, and an improved marginalization method is used to decompose the high-dimension matrix corresponding to the cost function, which binds the computational complexity and improves the computational efficiency.

Finally, experiments are conducted to compare SDPL-VIO and the other state-of-the-art VIO methods. The results of the European Robotics Challenge (EuRoC) datasets show that SDPL-VIO can consider both speed and accuracy.

The rest of this paper is organized as follows. In Section 2, the mathematical formulation is given. Next, the proposed system implementation is described in Section 3. The experimental results and analysis are shown in Section 4. Finally, a conclusion is given in Section 5.

2. Mathematical Formulation

2.1. Notations

We define $\{w\}$, $\{b\}$, $\{c\}$ as a world coordinate, body coordinate and camera coordinate, respectively. $\mathbf{R}_w^c$ and $\mathbf{p}_w^c$ are the rotation and translation from the world coordinate to the camera coordinate. $\mathbf{R}_c^b$ and $\mathbf{p}_c^b$ represent the extrinsic parameters, which can be calibrated in advance. $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix}$ is the 4×4 homogeneous transformation. $\mathbf{q}$ is the quaternion representation of $\mathbf{R}$, $\otimes$ represents the multiplication between two quaternions. $\lfloor * \rfloor_\times$ represents the skew symmetric matrix corresponding to the vector.

2.2. IMU Pre-Integration

The raw IMU measurements (angular velocity $\hat{\boldsymbol{\omega}}$ and acceleration $\hat{\mathbf{a}}$) are affected by gravity $\mathbf{g}^w$, bias $\mathbf{b}$ and noise $\mathbf{n}$ [10]:

$$\begin{aligned}\hat{\boldsymbol{\omega}} &= \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega \\ \hat{\mathbf{a}} &= \mathbf{a} + \mathbf{b}_\mathrm{a} + \mathbf{R}_w^b \mathbf{g}^w + \mathbf{n}_\mathrm{a}\end{aligned} \tag{1}$$

where, $\boldsymbol{\omega}$ and $\mathbf{a}$ are the real IMU measurements.

The IMU state propagation from the consecutive frame b_k to b_{k+1} can be given by:

$$\begin{aligned}\mathbf{p}_{b_{k+1}}^w &= \mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k - \tfrac{1}{2}\mathbf{g}^w \Delta {t_k}^2 + \mathbf{R}_{b_k}^w \boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\ \mathbf{v}_{b_{k+1}}^w &= \mathbf{v}_{b_k}^w - \mathbf{g}^w \Delta t_k + \mathbf{R}_{b_k}^w \boldsymbol{\beta}_{b_{k+1}}^{b_k} \\ \mathbf{q}_{b_{k+1}}^w &= \mathbf{q}_{b_k}^w \otimes \gamma_{b_{k+1}}^{b_k}\end{aligned} \tag{2}$$

where,

$$\begin{aligned}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \iint_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt^2 \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt \\ \gamma_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \tfrac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{\omega_t})\gamma_t^{b_k} dt\end{aligned} \tag{3}$$

where $\boldsymbol{\alpha}_{b_{k+1}}^{b_k}$, $\boldsymbol{\beta}_{b_{k+1}}^{b_k}$ and $\gamma_{b_{k+1}}^{b_k}$ are the pre-integrated measurements, and $\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor \boldsymbol{\omega} \rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}$.

2.3. Point Feature Projection

For a pin-hole camera model, the projection $\pi_c(*)$ from the camera coordinate $\mathbf{P}_c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T \in \mathbb{R}^3$ to the camera image plane $\mathbf{p} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ can be defined as:

$$\mathbf{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \pi_c(\mathbf{P}_c) = \frac{1}{z_c}\mathbf{K}\mathbf{P}_c = \frac{1}{z_c}\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{4}$$

where (f_x, f_y, c_x, c_y) represent camera intrinsic parameters.

2.4. Line Feature Projection

The Plücker coordinates [29] is used for line parameterization. A 3D line $\mathbf{L}$ representing the Plücker coordinates is constructed as $\mathbf{L} = \begin{bmatrix} \mathbf{n}^T & \mathbf{d}^T \end{bmatrix}^T$, where $\mathbf{n} \in \mathbb{R}^3$ represents the normal vector of the plane determined by the line and the coordinate origin, and $\mathbf{d} \in \mathbb{R}^3$ represents the line direction vector. According to the description in [30], the transformation of a 3D line $\mathbf{L}$ from the world coordinate $\mathbf{L}_w$ to the camera coordinate $\mathbf{L}_c$ can be defined as:

$$\mathbf{L}_c = \begin{bmatrix} \mathbf{n}_c \\ \mathbf{d}_c \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^c & \lfloor \mathbf{p}_w^c \rfloor_\times \mathbf{R}_w^c \\ 0 & \mathbf{R}_w^c \end{bmatrix} \mathbf{L}_w \tag{5}$$

and the projection from the camera coordinate to the camera image plane can be defined as:

$$\mathbf{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix} \mathbf{n}_c = \mathcal{K}\mathbf{n}_c \tag{6}$$

where $\mathcal{K}$ represents the projection matrix of line $\mathbf{L}_c$.

3. Proposed System Implementation

3.1. System Framework

The proposed system mainly includes two parts: front-end image tracking (measurements preprocessing and initialization), back-end nonlinear optimization, marginalization and loop closure detection, as shown in Figure 1. In the front end, the system extracts point and line features from the image, and carries out a motion estimation of adjacent image frames combined with IMU pre-integration. By aligning visual and inertial information, visual inertial joint initialization is performed to restore metric scale, and estimate inertial bias and gravity vector. Then, after judging whether the current frame is a keyframe according to the selection criteria, the system uses the semi-direct method to track keyframes and non-keyframes, respectively. In the back end, the system adopts a cost function to obtain the optimal state estimation by minimizing prior information, IMU residuals and visual re-projection errors, and uses improved marginalization to decompose the high-dimension matrix step-by-step, corresponding to the cost function, which optimizes the solution and improves the computational efficiency. The key processes are described in detail in the following sections.

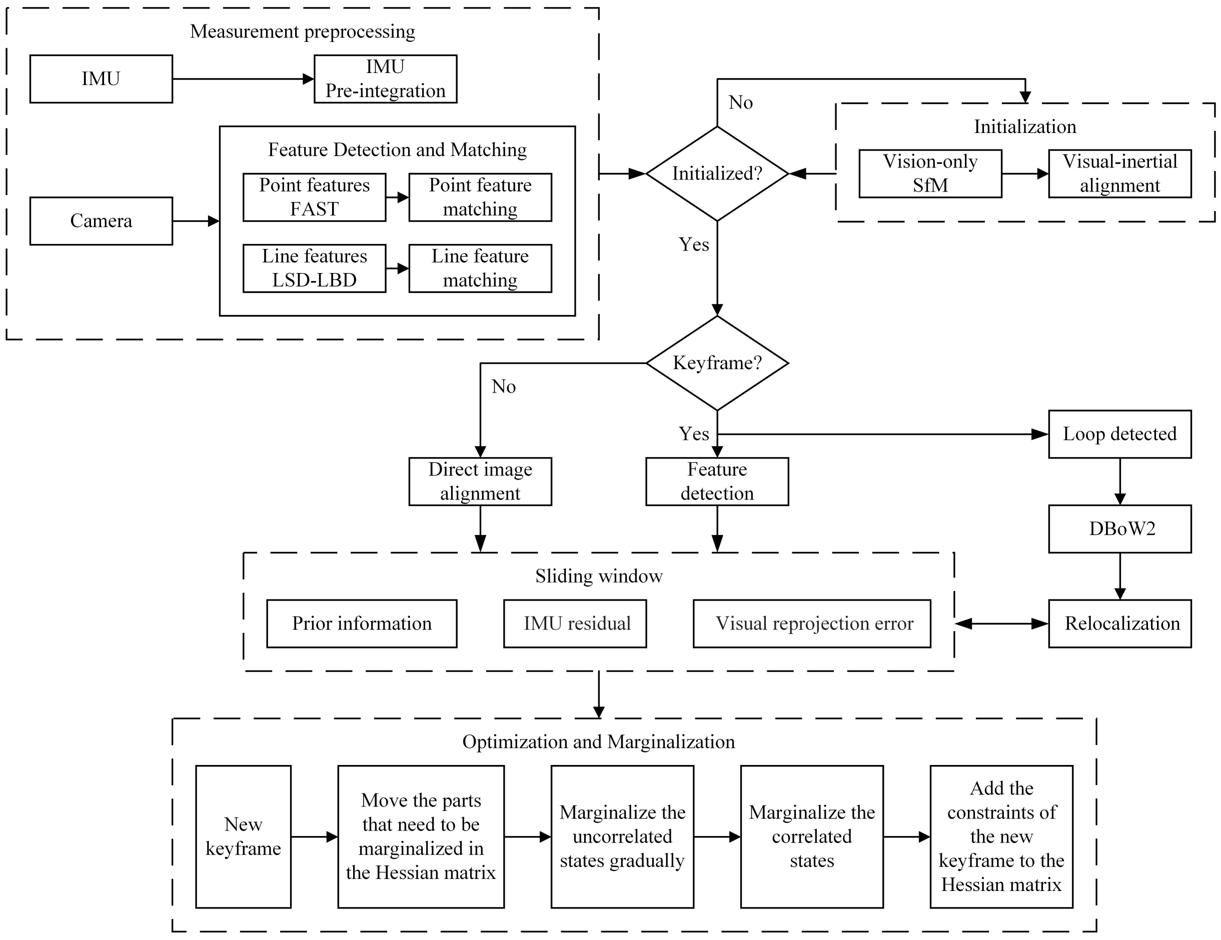
Measurement preprocessing
IMU
IMU Pre-integration
Feature Detection and Matching
Camera
Point features FAST
Point feature matching
Line features LSD-LBD
Line feature matching
No
Initialized?
Initialization
Vision-only SfM
Visual-inertial alignment
Yes
Keyframe?
No
Yes
Loop detected
Direct image alignment
Feature detection
DBoW2
Sliding window
Prior information
IMU residual
Visual reprojection error
Relocalization
Optimization and Marginalization
New keyframe
Move the parts that need to be marginalized in the Hessian matrix
Marginalize the uncorrelated states gradually
Marginalize the correlated states
Add the constraints of the new keyframe to the Hessian matrix

Figure 1. The system framework.

3.2. Front-End Tracking Based on Semi-Direct Methods

3.2.1. Visual-Inertial Initializaiton

We conducted measurement preprocessing, and completed the visual-inertial initialization by aligning the results of vision-only Structure from Motion (SFM) and IMU pre-integration. The system first performed this process to restore the metric scale and estimate the inertial bias and the gravity vector. The specific initialization process can be referred to in [11].

3.2.2. Keyframe Selection

After initialization, the system determines whether the current frame is a keyframe based on scene changes and IMU pre-integration. The overall output accuracy of the system depends largely on the quality of the keyframes that were inserted. The selection criteria for keyframes are as follows: First, the average parallax of tracked points between the current frame and the latest keyframe is beyond a certain threshold. Second, the number of tracked points goes below a certain threshold. Third, the translation calculated by IMU pre-integration exceeds a preset threshold after the latest keyframe is inserted. If any of the above criteria are met, the current frame is considered a keyframe.

3.2.3. Semi-Direct Method for Tracking

(a) For keyframes, we detected point features using the Accelerated Segment Test (FAST) [31] algorithm and tracked adjacent frames using an optical flow based on Kanade-Lucas-Tomasi (KLT) [32]. Then, we eliminated outliers by Random Sample Consensus (RANSAC) combined with the essential matrix model. Meanwhile we detected line segments using the Line Segment Detector (LSD) [33] algorithm, and matched them using Line Band Descriptors (LBD) [34] between the current frame and the reference frame. In addition, we also removed outliers for line-matching using geometric constraints.

(b) For non-keyframes, as shown in Figure 2, we first extracted the key points $\mathbf{p}$ from the last keyframe $\mathbf{I}_n$ in the sliding window, and used transformation to construct the matching 3D points $\mathbf{P}_w$ in the world coordinate. Then, we projected the 3D points onto the current frame $\mathbf{I}_{n+1}$, and obtained the pixel intensity residual $\delta\mathbf{I}(\mathbf{T},\mathbf{p})$ as:

$$\delta\mathbf{I}(\mathbf{T},\mathbf{p}) = \mathbf{I}_{n+1}(\pi_c(\mathbf{T}\cdot\mathbf{P}_w)) - \mathbf{I}_n(\mathbf{p}), \forall\mathbf{p}\in\mathcal{R} \tag{7}$$

where $\mathcal{R}$ is the image area formed by the key points $\mathbf{p}$. By minimizing the photometric error, the relative pose $\mathbf{T}_n^{n+1}$ can be calculated [15]:

$$\mathbf{T}_n^{n+1} = \arg\min_{\mathbf{T}} \iint_{\mathcal{R}} \rho[\delta\mathbf{I}(\mathbf{T},\mathbf{p})]d\mathbf{p} \tag{8}$$

where $\rho[\cdot] = \frac{1}{2}\|\cdot\|^2$.

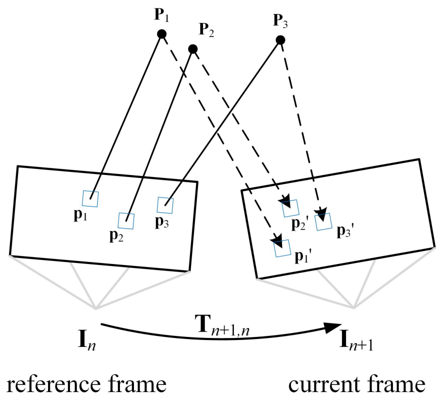

Figure 2. Direct image alignment.

However, the current frame pose, which was only obtained by Equation (8), is insufficiently accurate. Therefore, we need to find more co-visibility feature points between the current frame and the co-visibility keyframes in the sliding window. As shown in Figure 3, based on the pose solved by Equation (8), we projected the co-visible 3D points $\mathbf{P}_{wi}$ in the world coordinate onto the current frame. By minimizing the photometric error of co-visibility points, the corresponding 2D feature points $\mathbf{p}_i{}'$ in the current frame can be obtained:

$$\mathbf{p}_i{}' = \arg\min_{\mathbf{p}_i{}'} \frac{1}{2}\sum \left\|\mathbf{I}_{n+1}(\mathbf{p}_i{}') - \mathbf{I}_{n+1}(\pi_c(\mathbf{T}_w^c \cdot \mathbf{P}_{wi}))\right\|^2, \forall i \tag{9}$$

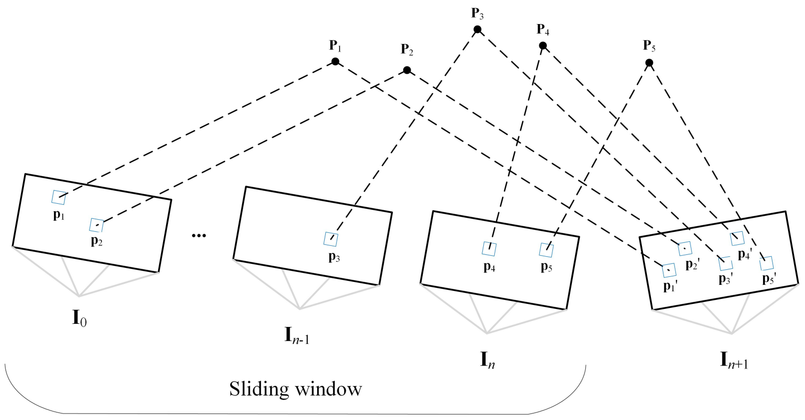

Figure 3. Pose refinement.

After feature-points-matching was completed, we again optimized the camera pose $\mathbf{T}_w^c$ to minimize the re-projection errors:

$$\mathbf{T}_w^c = \arg\min_{\mathbf{T}_w^c} \frac{1}{2}\sum_i \|\mathbf{p}_i - \pi_c(\mathbf{T}_w^c \cdot \mathbf{P}_{wi})\|^2 \tag{10}$$

3.3. Back-End Optimization

3.3.1. Sliding Window Formulation

The sliding window [28] limits the number of keyframes and prevents the number of poses and features from increasing over time, so that back-end optimization is always within a limited complexity. The full state variables in the sliding window are defined as:

$$\begin{aligned} \chi &= [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n, \mathbf{P}_{w1}, \mathbf{P}_{w2}, ..., \mathbf{P}_{wm}, \mathbf{L}_{w1}, \mathbf{L}_{w2}, ..., \mathbf{L}_{wo}] \\ \mathbf{x}_i &= [\mathbf{p}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{q}_{b_i}^w, \mathbf{b}_a, \mathbf{b}_g], i \in [1, n]. \end{aligned} \tag{11}$$

where x_i is the ith IMU state. $\mathbf{p}_{b_i}^w$, $\mathbf{v}_{b_i}^w$, and $\mathbf{q}_{b_i}^w$ represent position, velocity and orientation, respectively. $\mathbf{b}_a$ and $\mathbf{b}_g$ represent acceleration bias and gyroscope bias, respectively. $\mathbf{P}_{wi}(i \in [1, m])$ represent the point landmarks, and $\mathbf{L}_{wi}(i \in [1, o])$ are the 3D line representation formed by the Plücker coordinates. m and o are the number of point and line features in the sliding window, respectively.

The cost function, which simultaneously optimizes visual and IMU variables, is shown in Equation (12):

$$\min_{\chi}\left\{\|r_p - \mathbf{H}_p\chi\|^2 + \sum_{k\in\mathcal{B}} r_{\mathcal{B}}\|(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \chi)\|^2_{\Sigma_{b_{k+1}}^{b_k}} + \sum_{p\in\mathcal{P}} \rho(r_{\mathcal{P}}\|(\hat{\mathbf{z}}_p^{c_j}, \chi)\|^2_{\Sigma_p^{c_j}}) + \sum_{l\in\mathcal{L}} \rho(r_{\mathcal{L}}\|(\hat{\mathbf{z}}_l^{c_j}, \chi)\|^2_{\Sigma_l^{c_j}})\right\} \tag{12}$$

where $\{r_p, \mathbf{H}_p\}$ are the prior information from marginalization, $r_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \chi)$ are the IMU measurement residuals, $\mathcal{B}$ is the set of IMU states. $r_{\mathcal{P}}(\hat{\mathbf{z}}_p^{c_j}, \chi)$ and $r_{\mathcal{L}}(\hat{\mathbf{z}}_l^{c_j}, \chi)$ are the point and line feature re-projection errors, respectively. $\mathcal{P}$ and $\mathcal{L}$ are the set of observed point and line features, and ρ is the Cauchy loss function, which minimizes the influence of outliers. Therefore, the matching error terms can be expressed as follows:

(a) IMU measurement residual

According to Equations (1)–(3), the IMU measurement residuals for two consecutive frames can be calculated as follows:

$$r_{\mathcal{B}}(\widehat{\mathbf{z}}^{b_k}_{b_{k+1}}, \chi) = \begin{cases} \delta\boldsymbol{\alpha}^{b_k}_{b_{k+1}} \\ \delta\boldsymbol{\beta}^{b_k}_{b_{k+1}} \\ \delta\boldsymbol{\theta}^{b_k}_{b_{k+1}} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{cases} = \begin{cases} \mathbf{R}^{b_k}_w(\mathbf{p}^w_{b_{k+1}} - \mathbf{p}^w_{b_k} - \mathbf{v}^w_{b_k}\Delta t_k + \frac{1}{2}g^w \Delta t_k^2) - \widehat{\boldsymbol{\alpha}}^{b_k}_{b_{k+1}} \\ \mathbf{R}^{b_k}_w(\mathbf{v}^w_{b_{k+1}} - \mathbf{v}^w_{b_k} + g^w\Delta t_{ij}) - \widehat{\boldsymbol{\beta}}^{b_k}_{b_{k+1}} \\ 2[\mathbf{q}^{w^{-1}}_{b_k} \otimes \mathbf{q}^w_{b_{k+1}} \otimes (\widehat{\gamma}^{b_k}_{b_{k+1}})^{-1}]_{xyz} \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{\omega b_{k+1}} - \mathbf{b}_{\omega b_k} \end{cases} \tag{13}$$

where $[*]_{xyz}$ extracts the vector part of the quaternion.

(b) Point feature error term

As shown in Figure 4, the point feature re-projection error is represented as the distance between the observed projection position and currently estimated projection position of the 3D point, which can be defined as:

$$r_{\mathcal{P}}(\widehat{\mathbf{z}}^{c_j}_p, \chi) = \mathbf{p}^{c_j}_p - \pi_c\left(\mathbf{T}^{c_j}_w \cdot \mathbf{P}_{wp}\right) \tag{14}$$

where $\mathbf{p}^{c_j}_p$ is the observed point in camera frame j and $\mathbf{P}_{wp}$ is the matching 3D point.

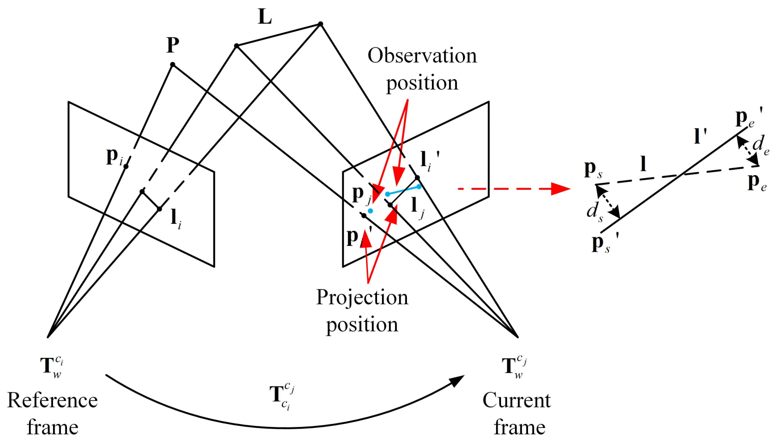

Figure 4. Illustration of the visual reprojection errors.

The Jacobian of the point feature re-projection error relative to the pose increment can be obtained by the chain rule:

$$\frac{\partial r_{\mathcal{P}}}{\partial \delta\xi} = \frac{\partial r_{\mathcal{P}}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \delta\xi} \tag{15}$$

with

$$\frac{\partial r_{\mathcal{P}}}{\partial \mathbf{p}} = -\begin{bmatrix} \frac{f_x}{z_c} & 0 & -\frac{f_x x_c}{{z_c}^2} \\ 0 & \frac{f_y}{z_c} & -\frac{f_y y_c}{{z_c}^2} \end{bmatrix} \tag{16}$$

$$\frac{\partial \mathbf{p}}{\partial \delta\xi} = \begin{bmatrix} 1 & 0 & 0 & 0 & z_c & -y_c \\ 0 & 1 & 0 & -z_c & 0 & -x_c \\ 0 & 0 & 1 & y_c & x_c & 0 \end{bmatrix} \tag{17}$$

(c) Line feature error term

According to Figure 4, the line feature re-projection error is represented as the distance from two endpoints ($\mathbf{p}_s = [u_1\ v_1\ 1]^T$ and $\mathbf{p}_e = [u_2\ v_2\ 1]^T$) of the matching line segment to the projecting line $\mathbf{l}$. Combining Equations (5) and (6), this can be calculated as below [29]:

$$r_{\mathcal{L}}(\hat{\mathbf{z}}_l^{c_i}, \chi) = \begin{bmatrix} d(\mathbf{p}_{s_l}^{c_i}, \mathbf{l}_l^{c_i}) \\ d(\mathbf{p}_{e_l}^{c_i}, \mathbf{l}_l^{c_i}) \end{bmatrix} \tag{18}$$

where,

$$d(\mathbf{p}, \mathbf{l}) = \frac{\mathbf{p}^T \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \tag{19}$$

The Jacobian line feature re-projection error relative to the pose increment can be obtained as follows:

$$\frac{\partial r_{\mathcal{L}}}{\partial \delta \xi} = \frac{\partial r_{\mathcal{L}}}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_c} \frac{\partial \mathbf{L}_c}{\partial \delta \xi} \tag{20}$$

with

$$\frac{\partial r_{\mathcal{L}}}{\partial \mathbf{l}} = \begin{bmatrix} \frac{u_1 l_2{}^2 - l_1 l_2 v_1 - l_1 l_3}{(l_1{}^2 + l_2{}^2)^{\frac{3}{2}}} & \frac{v_1 l_1{}^2 - l_1 l_2 u_1 - l_2 l_3}{(l_1{}^2 + l_2{}^2)^{\frac{3}{2}}} & \frac{1}{(l_1{}^2 + l_2{}^2)^{\frac{1}{2}}} \\ \frac{u_2 l_2{}^2 - l_1 l_2 v_2 - l_1 l_3}{(l_1{}^2 + l_2{}^2)^{\frac{3}{2}}} & \frac{v_2 l_1{}^2 - l_1 l_2 u_2 - l_2 l_3}{(l_1{}^2 + l_2{}^2)^{\frac{3}{2}}} & \frac{1}{(l_1{}^2 + l_2{}^2)^{\frac{1}{2}}} \end{bmatrix} \tag{21}$$

$$\frac{\partial \mathbf{l}}{\partial \mathbf{L}_c} = \begin{bmatrix} f_x & 0 & 0 & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 & 0 & 0 \\ -f_x c_x & f_x c_y & f_x f_y & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

$$\frac{\partial \mathbf{L}_c}{\partial \delta \xi} = \begin{bmatrix} -[\mathbf{R}_w^c \mathbf{n}_w]_\times - [\mathbf{p}_w^c]_\times [\mathbf{R}_w^c \mathbf{d}_w]_\times & -[\mathbf{R}_w^c \mathbf{d}_w]_\times \\ -[\mathbf{R}_w^c \mathbf{d}_w]_\times & 0 \end{bmatrix} \tag{23}$$

3.3.2. Improved Marginalization

Marginalization is performed to bound the computational complexity, and the illustration is shown in Figure 5. If the second latest frame is a keyframe, the system will marginalize the oldest frame, including the pose of the oldest frame and some observed visual landmarks. When the oldest frame has co-visibility with other keyframes in the sliding window, that is, they observe the same visual landmarks, marginalization will keep the constraint on other keyframes. Otherwise, the system will retain the IMU measurements attached to this non-keyframe but remove the visual measurements.

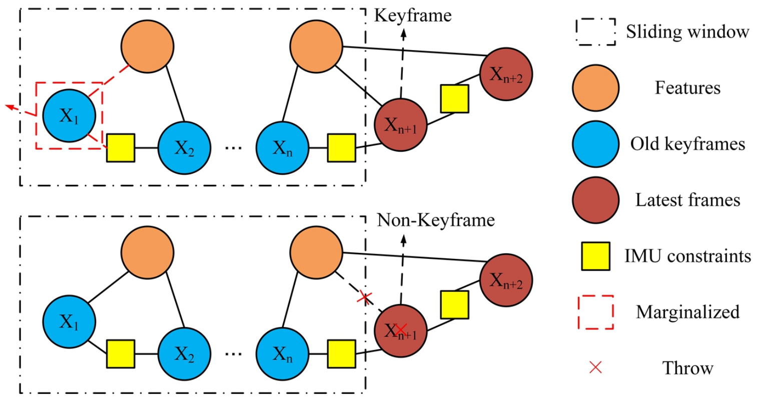

Figure 5. Illustration of marginalization.

This process is solved by the Gauss–Newton iterative method and defined in the form $\mathbf{H}\delta\mathbf{x} = \mathbf{b}$:

$$\begin{bmatrix} \mathbf{H}_m & \mathbf{H}_{mp} \\ \mathbf{H}_{mp}^T & \mathbf{H}_p \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}_m \\ \delta \mathbf{x}_p \end{bmatrix} = \begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_p \end{bmatrix} \tag{24}$$

where $\mathbf{b}_m$ represents the set of states that are to be marginalized and $\mathbf{b}_p$ represents the set of preserved states [23,24].

Through the Schur complement, $\delta\mathbf{x}_p$ can be obtained:

$$(\mathbf{H}_p - \mathbf{H}_{mp}^T\mathbf{H}_m^{-1}\mathbf{H}_{mp})\delta\mathbf{x}_p = \mathbf{b}_p - \mathbf{H}_{mp}^T\mathbf{H}_m^{-1}\mathbf{b}_m \tag{25}$$

which describes the basic marginalization. Thus, the states in $\mathbf{b}_m$ are marginalized together. Equation (25) requires calculating $\mathbf{H}_{mp}^T\mathbf{H}_m^{-1}\mathbf{H}_{mp}$, in which the computational complexity is $\mathcal{O}(i^3 + i^2 \times j + j^2 \times i)$, supposing that $\mathbf{H}_m$ and $\mathbf{H}_p$ have dimensions of $i \times i$ and $j \times j$.

The above basic marginalization method has high computational complexity, so we adopted the improved marginalization method to solve this problem. Firstly, we divided the state vector $\mathbf{x}_m$ which was be marginalized into two parts: one was uncorrelated states, containing visual landmarks, the other was correlated states, containing the pose of keyframes, velocity and IMU bias. Uncorrelated states were not related to each other, only to correlated states [24].

We described this process using an example, as shown in Figure 6. The states that are to be marginalized are represented by dashed lines, where $\mathbf{L}_{p_i} \in \mathbb{R}^3$ and $\mathbf{L}_{l_i} \in \mathbb{R}^6$ $(1 \leq i \leq 2)$ are points and line segments, respectively, $\mathbf{P}_j \in \mathbb{R}^6$ $(1 \leq j \leq 4)$ are the pose of keyframes, and $\mathbf{b}_k \in \mathbb{R}^9$ $(1 \leq k \leq 4)$ are velocity and IMU bias.

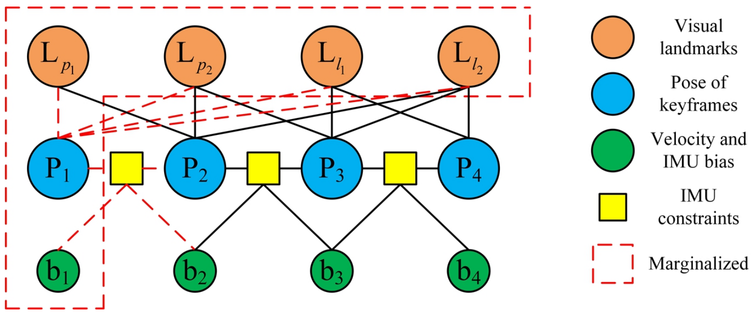

Figure 6. States in the sliding window.

Figure 7 shows the corresponding Hessian matrix. The state variable that is to be marginalized is $\mathbf{x}_m(\mathbf{L}_{p1}, \mathbf{L}_{p2}, \mathbf{L}_{l_1}, \mathbf{L}_{l_2}, \mathbf{P}_1, \mathbf{b}_1)$, while the state variable that is to be preserved is $\mathbf{x}_p(\mathbf{P}_2, \mathbf{b}_2, \mathbf{P}_3, \mathbf{b}_3, \mathbf{P}_4)$.

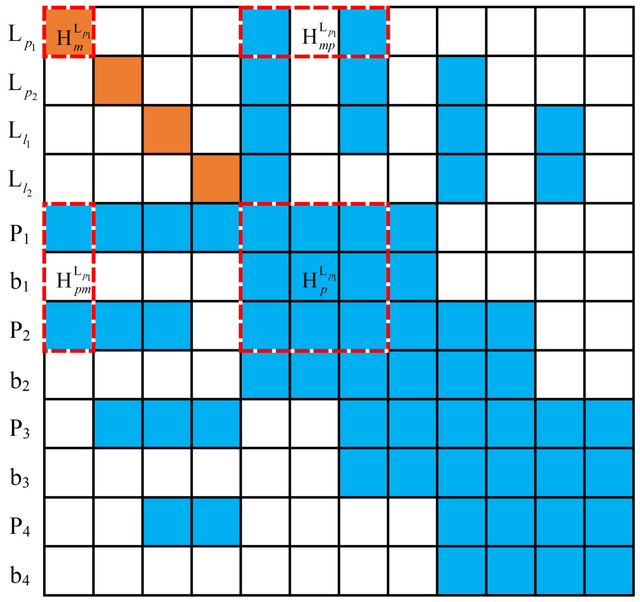

Figure 7. The Hessian matrix corresponding to Equation (24).

Firstly, uncorrelated states in $\mathbf{x}_m$ were marginalized, which only contain points and line segments ($\mathbf{L}_{p_1}$, $\mathbf{L}_{p_2}$, $\mathbf{L}_{l_1}$ and $\mathbf{L}_{l_2}$). The computational complexity for the marginalization of $\mathbf{L}_{p_1}$ in Figure 7 was mainly determined by $\mathbf{H}_{mp}^{\mathrm{L}_{p_1}}{}^{T}\mathbf{H}_{\mathrm{L}_{p_1}}{}^{-1}\mathbf{H}_{mp}^{\mathrm{L}_{p_1}}$, which is approximately $\mathcal{O}_{p_1} = \mathcal{O}(3^3 + 3^2 \times 21 + 3 \times 21^2)$. The maximum computational complexity will be max $\mathcal{O}_{p_1} = \mathcal{O}(3^3 + 3^2 \times 51 + 3 \times 51^2)$, supposing that $\mathbf{L}_{p_1}$ can be observed by $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{P}_4$. The Hessian matrix after the marginalization of $\mathbf{L}_{p_1}$ is shown in Figure 8.

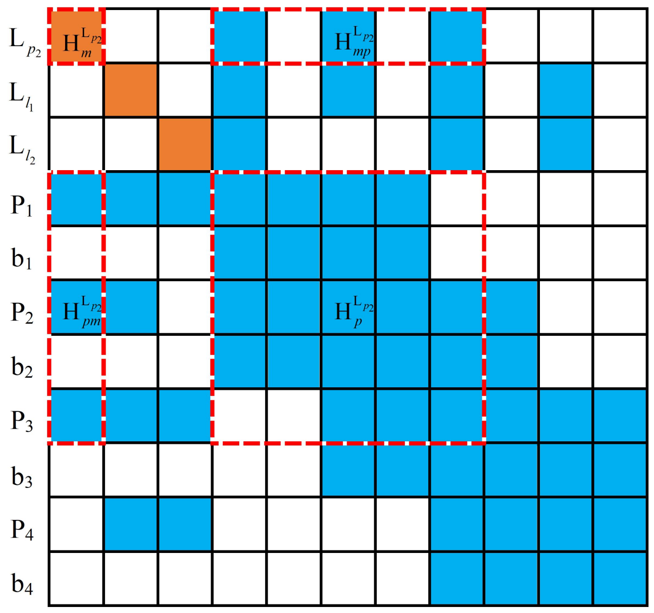

Figure 8. The Hessian matrix after marginalization of $\mathbf{L}_{p_1}$.

$\mathbf{L}_{p_2}$ will be marginalized after the marginalization of $\mathbf{L}_{p_1}$, which requires calculating $\mathbf{H}_{mp}^{\mathrm{L}_{p2}}{}^{T}\mathbf{H}_{\mathrm{L}_{p_2}}{}^{-1}\mathbf{H}_{mp}^{\mathrm{L}_{p2}}$, and the computational complexity is approximately $\mathcal{O}_{p_2} = \mathcal{O}(3^3 + 3^2 \times 36 + 3 \times 36^2)$.

The remaining $\mathbf{L}_{l_1}$ and $\mathbf{L}_{l_2}$ were marginalized in the same way, with a computational complexity of $\mathcal{O}_{l_1} = \mathcal{O}_{l_2} = \mathcal{O}(6^3 + 6^2 \times 51 + 6 \times 51^2)$. The Hessian matrix after the marginalization of uncorrelated states is shown in Figure 9. Then, correlated states that contain $\mathbf{P}_1$ and $\mathbf{b}_1$ in $\mathbf{x}_m$ were marginalized, by calculating $\mathbf{A}_p - \mathbf{A}_{mp}^T\mathbf{A}_m^{-1}\mathbf{A}_{mp}$, with a computational complexity of $\mathcal{O}_r = \mathcal{O}(15^3 + 15^2 \times 36 + 15 \times 36^2)$. Therefore, the total computational complexity will be $\mathcal{O} = \mathcal{O}_{p_1} + \mathcal{O}_{p_2} + \mathcal{O}_{l_1} + \mathcal{O}_{l_2} + \mathcal{O}_r$.

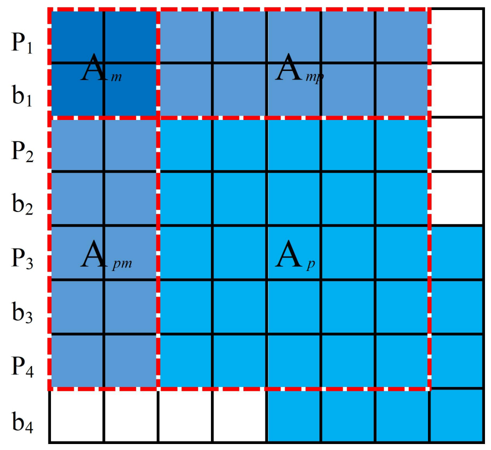

Figure 9. The Hessian matrix after the marginalization of all landmarks.

Through the above analysis, it is obvious that, compared with the basic marginalization method corresponding to Equation (25), the improved marginalization method greatly reduces the amount of computation.

3.3.3. Loop Closure Detection

For loop closure detection, we adopted Bags of Words (DBoW2) [35], a state-of-the-art bag-of words place recognition approach. The loop closure detection module will be activated when the current frame is selected as a keyframe. If a loop closure is detected, the drift accumulated during the exploration will be greatly reduced [10].

4. Experimental Results and Analysis

We compared SDPL-VIO with the state-of-the-art methods: SVO [17], PCSD-VIO [22], PL-VIO [11] and VINS-Mono [10] on the EuRoC datasets [36]. The EuRoC datasets were collected by various sensors installed on the MAV, and divided into three series of flight scenarios, including a series of motion sequences ranging from easy to difficult. First, the comparison results of the accuracy and robustness performance based on the datasets are shown in Section 4.1. Then, to demonstrate the computation performance of SDPL-VIO, the time usage statistics results are illustrated in Section 4.2. Finally, the loop-closure detection capability of SDPL-VIO is evaluated in Section 4.3.

4.1. Accuracy and Robustness Performance

We implemented the above methods on the EuRoC datasets. In order to intuitively reflect the tracking effect of our proposed semi-direct method, and because PL-VIO lacks the loop-closure module, we disabled the loop-closure module of the above methods. A comparison of root–mean–square error (RMSE) on the EuRoC datasets is shown in Table 1, and their histograms are also provided, as shown in Figure 10. Table 1 shows that SDPL-VIO works robustly and accurately, and other methods also work well. Compared with point features, the combination of point and line features can strengthen the constraints between images, are insensitive to illumination changing environment, deal with the dynamic environment, and reduce the error caused by fast rotation motion. Thus, due to the introduction of line features, the accuracy of SDPL-VIO is better than SVO, PCSD-VIO and VINS-Mono, which only uses point features for tracking. In easy sequences such as MH01 and V101, which have rich features, good illumination, and slow motion, the feature-based method can extract a high number of point and line features. In such environments, the quality of tracking is about the same, so the accuracy of SDPL-VIO is comparable to PL-VIO. In difficult sequences, such as MH05, V203 with motion blur, fast motion, low texture, etc., the feature-based method faces challenges due to the lack of sufficient visual features. Our proposed semi-direct method combines the excellent performance of direct methods in low-texture environments. The feature-based method provides an accurate initialization state and generates keyframes that provide good priors for the direct method, while the direct method uses direct image alignment to move the frame very close to its final pose, while using a refinement step to reduce the pose estimation error. Therefore, SDPL-VIO performs relatively well in difficult sequences.

Table 1. RMSE (m) of the five methods.

Sequence	SDPL-VIO	PCSD-VIO	PL-VIO	VINS-Mono	SVO
MH01	0.13	0.16	0.14	0.16	0.17
MH02	0.15	0.17	0.15	0.19	0.27
MH03	0.18	0.22	0.19	0.20	0.43
MH04	0.31	0.35	0.33	0.36	1.36
MH05	0.23	0.28	0.25	0.30	0.51
V101	0.06	0.10	0.07	0.09	0.20
V102	0.10	0.11	0.08	0.11	0.47

Table 1. *Cont.*

Sequence	SDPL-VIO	PCSD-VIO	PL-VIO	VINS-Mono	SVO
V103	0.13	0.18	0.14	0.20	N/A
V201	0.08	0.09	0.07	0.09	0.30
V202	0.11	0.14	0.12	0.16	0.47
V203	0.22	0.26	0.24	0.28	N/A

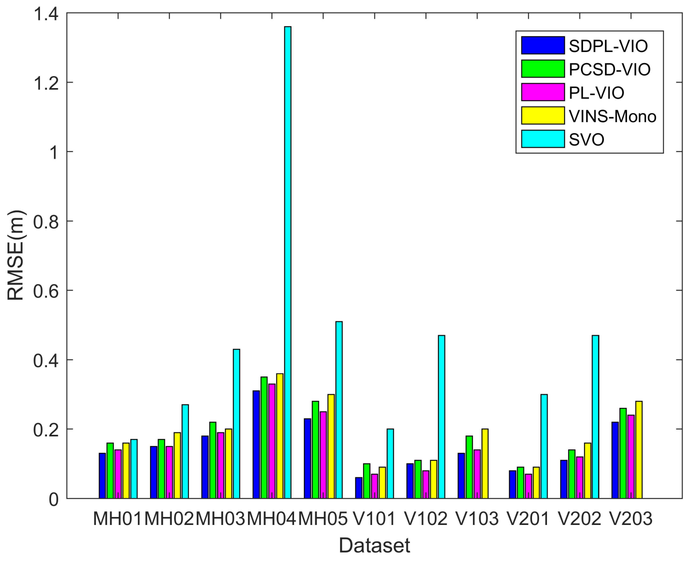

Figure 10. RMSE (m) of the five methods.

Figure 11 shows a comparison of the trajectories obtained by SDPL-VIO, PCSD-VIO, PL-VIO and VINS-Mono on the MH01, MH05 and V202 datasets, respectively. The ground truth is represented by the red line, while the results of SDPL-VIO, PCSD-VIO, PL-VIO and VINS-Mono are represented by the blue line, the green line, the purple line and the yellow line, respectively.

Figure 12 shows the translation errors on the MH01, MH05 and V202 datasets, in which the blue represents SDPL-VIO, the green represents PCSD-VIO, the purple represents PL-VIO, and the yellow represents VINS-Mono. It can been seen that, in the MH01 sequence, due to the rich scene features and good illumination conditions, there is not much difference in the accuracy of the four methods. The maximum error of SDPL-VIO is 0.37 m in the sequence of MH01, while that of PCSD-VIO is 0.41 m, that of PL-VIO is 0.41 m and that of VINS-Mono is 0.40 m, respectively. The MH05 dataset includes fast motion and large illumination changes. The combination of point and line features is more robust to fast rotation in the trajectory, and the semi-direct method is more adaptable to low-texture environments, so SDPL-VIO has the highest accuracy. PCSD-VIO and VINS-Mono only extract point features, and they struggle to extract corner points with large grayscale differences from surrounding pixel blocks. Therefore, the number of effective feature points is reduced, and the accuracy is the lower than the other two methods. The maximum error of SDPL-VIO is 0.73 m in the sequence of MH05, while that of PCSD-VIO is 0.85 m, that of PL-VIO is 0.78 m and that of VINS-Mono is 0.90 m, respectively. For the V202 dataset, SDPL-VIO still performs better than the other three. The maximum error of SDPL-VIO is 0.33 m, while that of PCSD-VIO is 0.37 m, that of PL-VIO is 0.36 m and that of VINS-Mono is 0.42 m, respectively. From Figures 11 and 12, it can been seen that, due to the semi-direct method tracking point and line features, SDPL-VIO has a better performance than the other three methods in challenging sequences, such as fast motion, large illumination changes or poorly textured environments, etc.

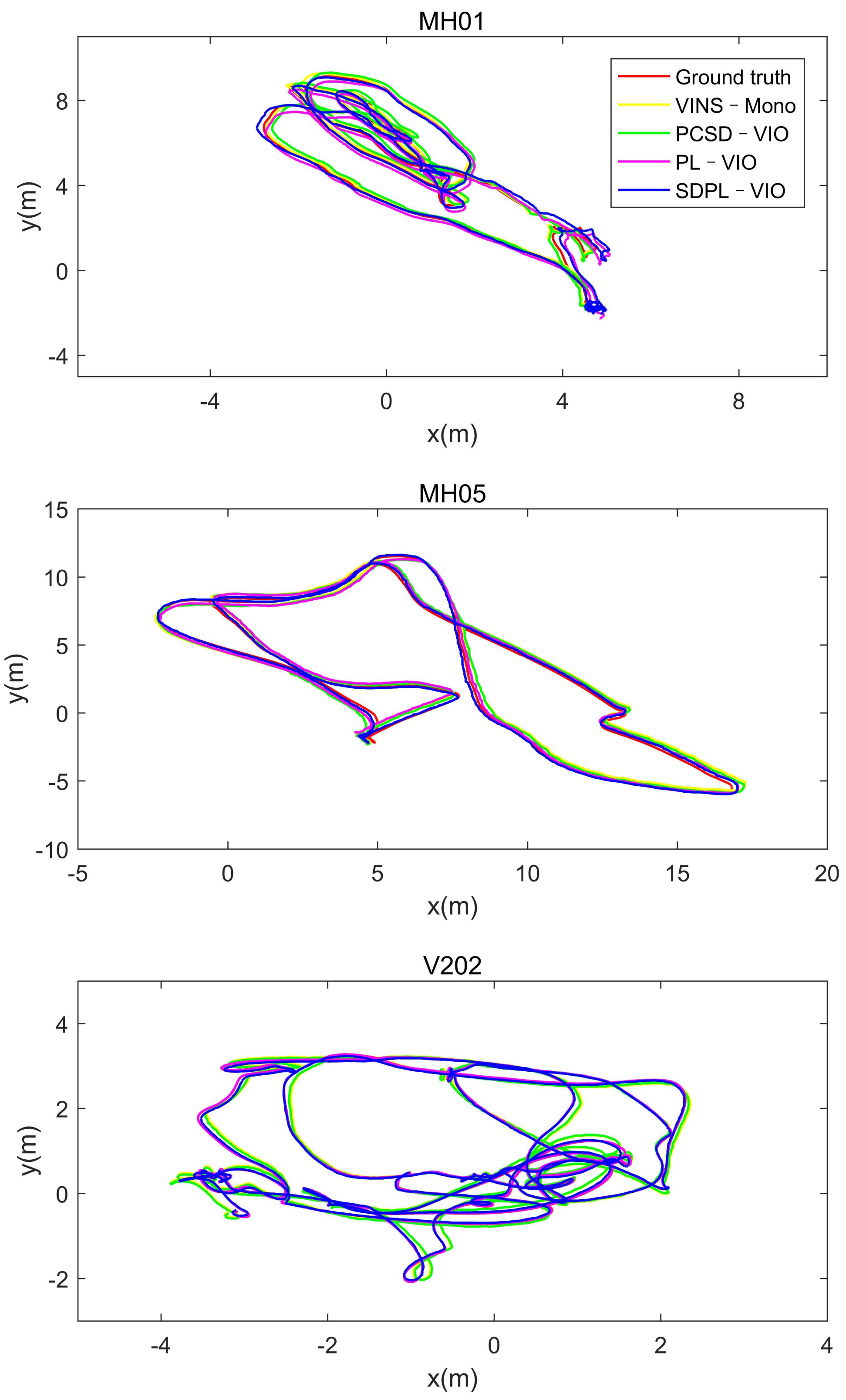

Figure 11. The comparison of trajectories on the MH01, MH05 and V202 datasets.

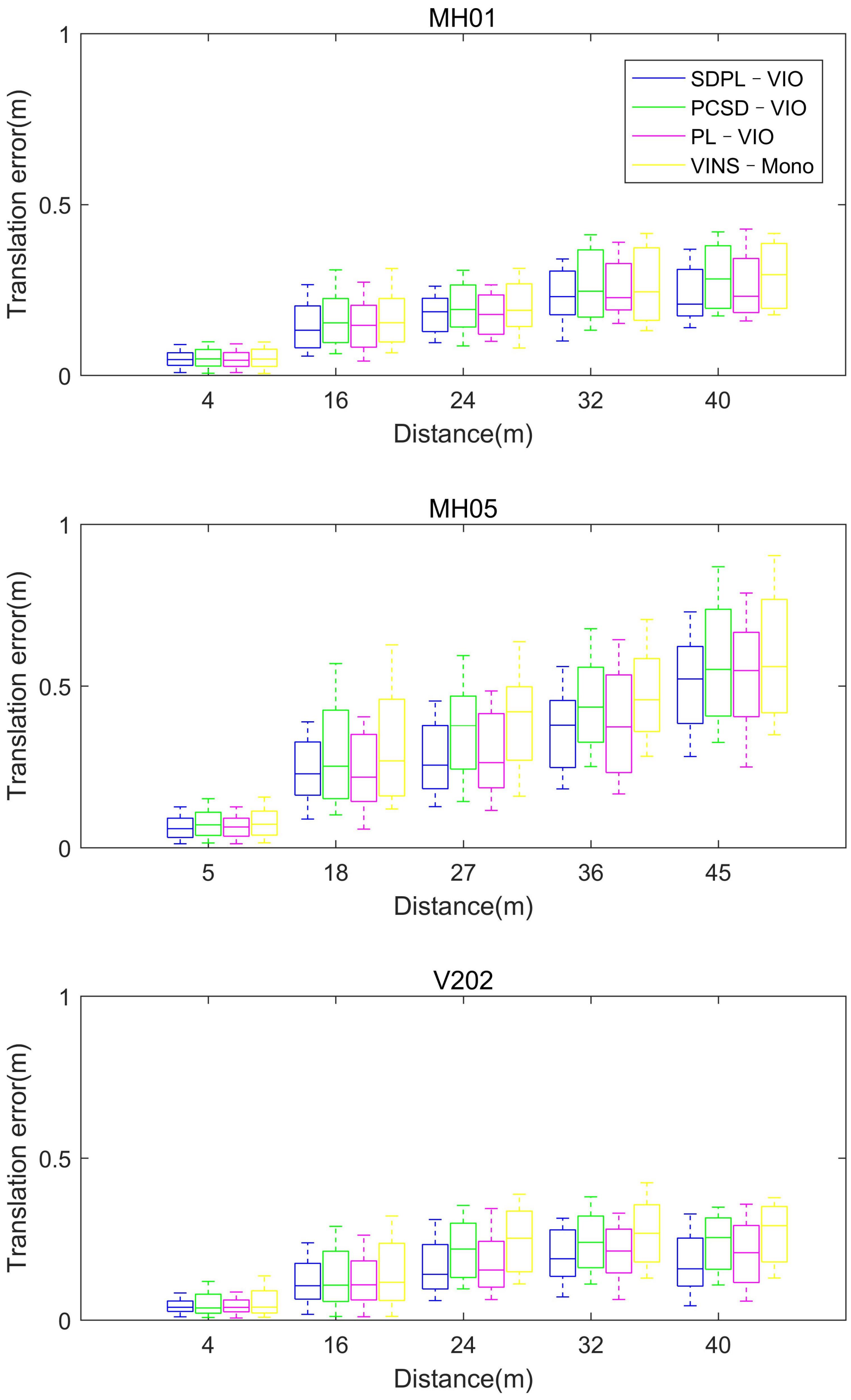

Figure 12. The comparison of translation errors on the MH01, MH05 and V202 datasets.

4.2. Computational Performance

To evaluate the computational performances of SDPL-VIO, the average times taken for tracking and marginalization were measured and analyzed.

4.2.1. Average Time for Tracking

Firstly the computational performances when tracking the image were analyzed to compare SDPL-VIO and PL-VIO, and the comparison results are shown in Table 2. PL-VIO simultaneously extracts point and line features for each frame, which is very time-consuming. SDPL-VIO uses the semi-direct method for tracking, and takes much less time than PL-VIO. This is because non-keyframes account for the majority of the tracked features and the system uses the direct method, with the advantage of rapidity when tracking non-keyframes in the front end, and has no need to extract image features and calculate descriptors. This strategy effectively reduces the average front-end tracking time. Taking V201 as an example, as shown in Figure 13, according to the keyframe selection strategy, the number of selected keyframes is 127, accounting for about 23% of the total frames, while 77% of the frames were selected to be non-keyframes. However, the time needed to track keyframes accounted for 69% of the total time, while the time needed to track non-keyframes accounted for only 31%. Although the tracking time was reduced, the accuracy of SDPL-VIO was not significantly reduced; this was still comparable with PL-VIO, and even higher in some environments, as shown in Section 4.1.

Table 2. Average time (ms) spent tracking an image.

Sequence	SDPL-VIO	PL-VIO
MH01	28.54	84.40
MH02	33.55	88.05
MH03	27.51	85.73
MH04	30.45	83.18
MH05	30.10	83.10
V101	29.65	82.52
V102	29.48	83.35
V103	32.81	87.61
V201	30.15	81.24
V202	31.65	89.76
V203	33.56	90.40

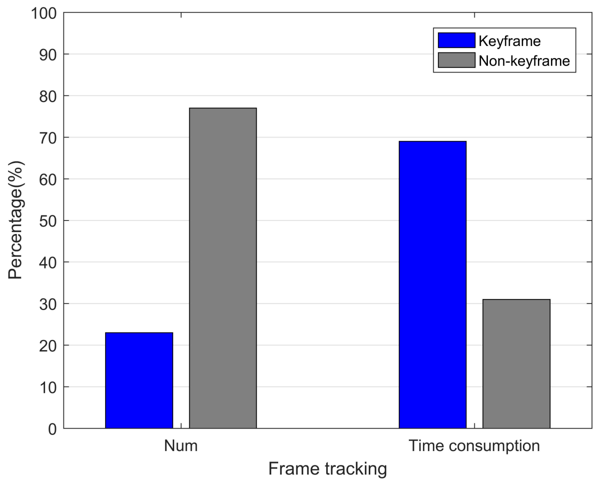

Figure 13. The ratio of the number and tracking time between keyframes and non-keyframes on the V201 dataset.

4.2.2. Average Time for Marginalization

Marginalization is another time-consuming aspect of the back-end, so the average time consumption was also analyzed. Taking V102 as an example, as explained in Section 3.3.2, when the back end of SDPL-VIO used the basic marginalization method, one-step marginalization was carried out, resulting in a very large amount of computation, and the average time needed for marginalization was 39 ms. When the back end used the improved marginalization method, the high-dimension matrix corresponding to the cost function was decomposed step by step. Therefore, the amount of computation was significantly reduced, and the average time needed for marginalization was 16 ms.

4.3. Loop Closure Detection Evaluation

The most obvious advantage of the feature-based method compared with the direct method is that the feature-based method could be used for loop closure detection, which can reduce the drift accumulated during the long-term operation. Therefore, the loop-closure detection ability of SDPL-VIO was finally evaluated to verify the integrity and reliability of the system. From the comparison results on the MH03 dataset, as shown in Figure 14, the accuracy of SDPL-VIO with loop closure detection is clearly shown to improve.

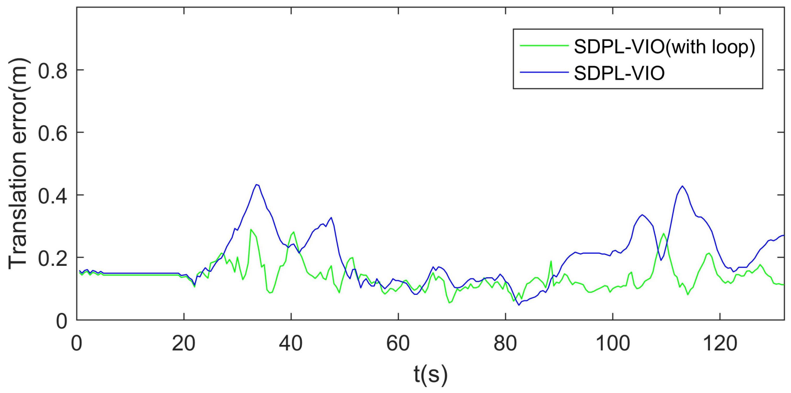

Figure 14. The comparison of translation errors on the MH03 dataset.

The analysis of multiple experiments shows that SDPL-VIO can consider both speed and accuracy, which increases the operation efficiency without reducing the operation precision, and is superior to other methods in terms of comprehensive performance.

5. Conclusions

In this paper, a novel, semi-direct, point-line, visual inertial odometry for MAVs was proposed, which extracts point-line features from the image and uses the semi-direct method to track keyframes and non-keyframes. The proposed method also adopts the sliding window strategy and uses the improved marginalization method to decompose the high-dimensional matrix step by step, according to the cost function, to optimize the solution. Experiments on the EuRoC datasets show that the accuracy and real-time performance of SDPL-VIO is better than that of other state-of-the-art VIO methods, especially in challenging datasets containing fast motion, large illumination changes or poorly textured environments. The SDPL-VIO performance, in terms of accuracy and efficiency, validated that it is suitable for navigational uses in MAVs with low-cost sensors. In future work, we aim to adopt a faster line detector that could be more conducive to continuous and real-time tracking.

Author Contributions: Conceptualization, B.G., B.L. and C.T.; Formal analysis, B.G. and C.T.; Investigation, B.G.; Methodology, B.G. and C.T.; Project administration, B.L.; Resources, B.L.; Software, B.G. and C.T.; Supervision, B.L.; Validation, B.G.; Writing—original draft, B.G.; Writing—review & editing, B.G. and C.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Natural Science Foundation of China under Grant 62171735, 62271397, 62173276, 62101458, 61803310 and 61801394, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2022GY-097,2021JQ-122 and 2021JQ-693, and in part by China Postdoctoral Science Foundation under Grant 2020M673482 and 2020M673485.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

MAVs	Micro Aerial Vehicles
IMU	Inertial Measurement Unit
VIO	Visual Inertial Odometry
VIN	Visual Inertial Navigation
OKVIS	Keyframe-based visual-inertial SLAM using nonlinear optimization
BRISK	BRISK: Binary robust invariant scalable keypoints
ORB	Oriented FAST and Rotated BRIEF
ORB-SLAM2	ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras
PL-SLAM	PL-SLAM: Real-time monocular visual SLAM with points and lines
DSO	Direct Sparse Odometry
VINS-Mono	VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator
PL-VIO	PL-VIO: Tightly-Coupled Monocular Visual–Inertial Odometry Using Point and Line Features
SVO	SVO: Fast semi-direct monocular visual odometry
PL-SVO	PL-SVO: Semi-direct Monocular Visual Odometry by combining points and line segments
SVL	Semi-direct monocular visual and visual-inertial SLAM with loop closure detection
PCSD-VIO	PC-SD-VIO: A constant intensity semi-direct monocular visual-inertial odometry with online photometric calibration
EuRoC	the European Robotics Challenge
SFM	Structure from Motion
FAST	the Accelerated Segment Test
KLT	Kanade-Lucas-Tomasi
RANSAC	Random Sample Consensus
LSD	Line Segment Detector
LBD	Line Band Descriptors
DBoW2	Bags of Words
RMSE	root-mean-square error

References

1. Citroni, R.; Di Paolo, F.; Livreri, P. A Novel Energy Harvester for Powering Small UAVs: Performance Analysis, Model Validation and Flight Results. *Sensors* **2019**, *19*, 1771. [CrossRef]
2. Delmerico, J.; Scaramuzza, D. A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 2502–2509.
3. Cheng, J.; Zhang, L.; Chen, Q.; Hu, X.; Cai, J. A review of visual SLAM methods for autonomous driving vehicles. *Eng. Appl. Artif. Intell.* **2022**, *114*, 104992. [CrossRef]
4. Mourikis, A.I.; Roumeliotis, S.I. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy, 10–14 April 2007; pp. 3565–3572.
5. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Robot. Res.* **2014**, *34*, 314–334. [CrossRef]
6. Bloesch, M.; Omari, S.; Hutter, M.; Siegwart, R. Robust visual inertial odometry using a direct EKF-based approach. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 298–304.
7. 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. [CrossRef]
8. 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 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508.
9. Mur-Artal, R.; Tardós, J.D. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robot. Autom. Lett.* **2017**, *2*, 796–803. [CrossRef]

10. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
11. He, Y.; Zhao, J.; Guo, Y.; He, W.; Yuan, K. PL-VIO: Tightly-Coupled Monocular Visual–Inertial Odometry Using Point and Line Features. *Sensors* **2019**, *18*, 1159. [CrossRef]
12. Duan, R.; Paudel, D.P.; Fu, C.; Lu, P. Stereo Orientation Prior for UAV Robust and Accurate Visual Odometry. *IEEE/ASME Trans. Mechatronics* **2022**, 1–11. [CrossRef]
13. Engel, J.; Schps, T.; Cremers, D. *LSD-SLAM: Large-Scale Direct Monocular SLAM;* Springer: Cham, Switzerland, 2014.
14. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625. [CrossRef]
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 (ICRA), Hong Kong, China, 31 May–7 June 2014.
16. Gomez-Ojeda, R.; Briales, J.; Gonzalez-Jimenez, J. PL-SVO: Semi-direct Monocular Visual Odometry by combining points and line segments. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 4211–4216.
17. Forster, C.; Zhang, Z.; Gassner, M.; Werlberger, M.; Scaramuzza, D. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. *IEEE Trans. Robot.* **2017**, *33*, 249–265. [CrossRef]
18. Li, S.; Zhang, T.; Gao, X.; Wang, D.; Xian, Y. Semi-direct monocular visual and visual-inertial SLAM with loop closure detection. *Robot. Auton. Syst.* **2019**, *112*, 201–210. [CrossRef]
19. Lee, S.H.; Civera, J. Loosely-Coupled Semi-Direct Monocular SLAM. *IEEE Robot. Autom. Lett.* **2019**, *4*, 399–406. [CrossRef]
20. Dong, X.; Cheng, L.; Peng, H.; Li, T. FSD-SLAM: A fast semi-direct SLAM algorithm. *Complex Intell. Syst.* **2021**, *8*, 1823–1834. [CrossRef]
21. Luo, H.; Pape, C.; Reithmeier, E. Hybrid Monocular SLAM Using Double Window Optimization. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4899–4906. [CrossRef]
22. Liu, Q.; Wang, Z.; Wang, H. PC-SD-VIO: A constant intensity semi-direct monocular visual-inertial odometry with online photometric calibration. *Robot. Auton. Syst.* **2021**, *146*, 103877. [CrossRef]
23. Usenko, V.; Demmel, N.; Schubert, D.; Stueckler, J.; Cremers, D. Visual-Inertial Mapping with Non-Linear Factor Recovery. *IEEE Robot. Autom. Lett. (RA-L) Int. Intell. Robot. Autom. (ICRA)* **2020**, *5*, 422–429. [CrossRef]
24. Xiao, J.; Xiong, D.; Yu, Q.; Huang, K.; Lu, H.; Zeng, Z. A Real-Time Sliding Window based Visual-Inertial Odometry for MAVs. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4049–4058. [CrossRef]
25. Guan, Q.; Wei, G.; Wang, L.; Song, Y. A Novel Feature Points Tracking Algorithm in Terms of IMU-Aided Information Fusion. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5304–5313. [CrossRef]
26. Xu, C.; Liu, Z.; Li, Z. Robust Visual-Inertial Navigation System for Low Precision Sensors under Indoor and Outdoor Environments. *Remote Sens.* **2021**, *13*, 772. [CrossRef]
27. Stumberg, L.v.; Cremers, D. DM-VIO: Delayed Marginalization Visual-Inertial Odometry. *IEEE Robot. Autom. Lett.* **2022**, *7*, 1408–1415. [CrossRef]
28. Sibley, G.; Matthies, L.; Sukhatme, G. Sliding window filter with application to planetary landing. *J. Field Robot.* **2010**, *27*, 587–608. [CrossRef]
29. Bartoli, A.; Sturm, P. The 3D line motion matrix and alignment of line reconstructions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; pp. 287–292.
30. Bartoli, A.; Sturm, P. Structure-From-Motion Using Lines: Representation, Triangulation and Bundle Adjustment. *Comput. Vis. Image Underst.* **2005**, *100*, 416–441. [CrossRef]
31. 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. [CrossRef]
32. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the Imaging Understanding Workshop, Vancouver, BC, Canada, 24–28 August 1981; pp. 121–130.
33. Gioi, R.; 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. [CrossRef]
34. 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. [CrossRef]
35. Galvez-Lopez, D.; Tardos, J.D. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197. [CrossRef]
36. 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. [CrossRef]