Research of Binocular Visual Inertial Algorithm Based on Point and Line Features

Yipeng Zhou1,2, Maohai Li1,2, Guowei Shao1,2
1College of Mechanical and Electrical Engineering, Soochow University, Suzhou, Jiangsu, 215131, China
2Key Laboratory of Jiangsu Robotics and Microsystems Research Center, Suzhou, Jiangsu, 215131, China
First author’s e-mail: 20194229018@stu.suda.edu.cn

Abstract: To solve the problem of poor performance of the binocular visual inertial odometer VINS-Fusion in scenes with low texture and large luminosity changes, a binocular visual inertial odometer PLVINS-Fusion is designed that integrates line feature measurement information, which use line features to easy to extract in low-texture scenes, and have the advantage of more robust tracking performance in scenes with large luminosity changes. Point and line features are extracted in the front-end visual extraction at the same time, and line feature residuals are added to the back-end nonlinear optimization, construct a bag-of-words model combining point and line features in the loop detection module. On this basis, a real-time photometric calibration algorithm is adopted to jointly optimize the exposure time, the camera response function and the vignetting factor, and the stability of KLT optical flow tracking is improved by correcting the image brightness. Experiments on benchmark dataset show that the optimized algorithm has higher robustness and effectively improves the positioning accuracy, and meets the real-time performance requirement.

1. Introduction
In recent years, mobile robots have been widely used in manufacturing, Internet, artificial intelligence and other fields. SLAM technology, as a key technology to assist robots in autonomous localization in unknown environments, has become a research hotspot in the field of mobile robots [1]. Visual inertial odometry has become an important direction for SLAM research because it can effectively fuse the data provided by cameras and IMU sensors and improve the robustness of the system. The binocular visual inertial odometer obtains depth information through binocular stereo matching and can estimate depth both at rest and in motion, with higher robustness compared to monocular cameras and unlike RGB-D cameras [2] which are not affected by infrared structured light.

Visual inertial odometry can be divided into two methods based on state estimation, filter-based and optimization-based [3]. The filter-based methods usually use Kalman filters to fuse the camera and IMU measurements, construct an error function through the equations of motion and the equations of observation and minimize the error function for bit-pose estimation [4]. The main algorithms that have applied the filter-based method are MSCKF-VIO (Multi-State Constraint Kalman Filter-Visual- Inertial Odometry) [5], ROVIO (Robust Visual-Inertial Odometry) [6] and OpenVINS [7]. Optimization-based methods usually use joint optimization of visual and IMU measurement residuals for graph optimization at the back-end, which can achieve higher accuracy while consuming more computational resources [8].

Optimization methods can be divided into point-based and line-based methods depending on the type
of visual features extracted from the front-end. Point features have been adopted by many visual inertial odometry methods due to their simple description and ease of extraction, such as ORBSLAM3 (Oriented FAST and Rotated BRIEF SLAM3) [9], VINS-Fusion (Visual-Inertial Navigation Systems-Fusion), etc. The classical algorithms all use the extraction of point features at the front-end. However, point features are difficult to detect in low-texture or luminosity transformed scenes, with low localization accuracy and loopback false detection [10-11]. PL-SLAM [12] is the first binocular vision SLAM algorithm to fuse point and line features based on ORBSLAM [13]. PL-VIO [14] is the first monocular visual inertial odometry algorithm based on VINS-Mono [15], which uses point-line features tightly coupled with inertial measurement information for higher accuracy. PL-VINS adds a loopback based on point features to PL-VIO. The above research work suffers from the disadvantages of insufficient robustness in the face of light transformation scenarios and difficulty in detecting loopback when point features are scarce. Studies have shown that online photometric calibration of images can effectively improve the algorithm's robustness to light-changing scenes. In 2018, Jakob Engel proposed a real-time photometric calibration method, which was applied to the visual odometer method DSO [16] to effectively improve the accuracy of the algorithm.

VINS-Fusion is a classic binocular visual inertial mileage calculation method. However, in scenes with low texture or luminosity transformations, VINS-Fusion is unable to use visual information for bit-pose estimation because of the difficulty in detecting point features, and even loopback false detection occurs, resulting in poor performance. In this paper, we introduce line features in the framework of VINS-Fusion, incorporate an online photometric calibration algorithm, and propose a binocular visual inertial odometry method PLVINS-Fusion with a combination of point and line features. The main contributions of this paper are as follows:

1. On the basis of VINS-Fusion, line features are merged to construct a binocular visual inertial odometer framework based on point line features.
2. An online photometric calibration algorithm is used to improve the stability of KLT optical flow tracking and line feature tracking by correcting the image brightness.
3. Construct a bag-of-words model with a combination of point and line features, while considering descriptors of point and line features to improve the loopback detection effect.
4. Experiments on benchmark dataset show that the improved algorithm has higher positioning accuracy and robustness while ensuring real-time performance.

2. Algorithm overview

The binocular vision inertial odometry calculation method PLVINS-Fusion proposed in this paper adds line feature information on the basis of VINS-Fusion, so that the line features play a role in the front-end measurement preprocessing, back-end sliding window optimization and loop closure parts, and adds a photometric calibration module in the front-end to improve the positioning accuracy of the algorithm in the photometric transformation environment, the system algorithm flow is shown in Figure 1. The improved parts are indicated by blue marks.
Fig 1. Flow chart of the proposed algorithm

(1) Front-end measurement processing: The front-end acquires data information from the raw measurement data of the camera and IMU. The image is first photometric calibrated to facilitate subsequent point and line feature extraction and tracking. Point and line features are detected in two separate threads, with Shi-Tomasi corner points and LSD (Line Segment Detector) line segments extracted in the image when a new image frame arrives, respectively. Point features are matched by the KLT (Kanade-Lucas-Tomasi Tracking) optical flow method for left-right eye matching and front-back frame matching. Line features are matched using LBD (Line Band Descriptor) descriptors for left-right eye matching and front-back frame matching. After successful left and right eye matching, stereo matching is used to obtain depth information. During optimization, the pose of subsequent cameras is constrained by pre-integrating the IMU measurements and aligning them with the image frames. The visual and IMU information alignment is considered a successful initialization and triggers the non-linear optimization thread.

(2) Back-end sliding window optimization: finding the optimal state, velocity, spatial points and lines, acceleration and gyroscope bias in a fixed size sliding window. The window size is maintained by jointly minimizing the residual function via the LM (Levenburg-Marquard) algorithm, using marginalized sliding window oldest frames to ensure the sustainability of the algorithm.

(3) Loop closure: After sliding window optimization, the current frame is determined to be a key frame based on whether the parallax between the current frame and the last key frame exceeds a threshold value. After the current frame is confirmed as a key frame, the loopback detection thread is activated to match the descriptors of point and line features through the dictionary. After the loopback is detected, the system treats the loopback candidate frame as the correct loopback frame and repositions it, aligning the current sliding window maintained by the binocular vision inertial odometer with the past poses, which has the effect of reducing the cumulative error.

3. Algorithm flow

3.1 Image photometric calibration
When there are strong exposure changes between images, the KLT optical flow tracking algorithm based on the assumption of constant gray level will fail. This is because the camera's automatic exposure causes the overall image luminosity to change. To solve this problem, the photometric calibration algorithm is used to perform photometric compensation on the image, thereby improving the stability of KLT optical flow tracking.
The reflection from a scene illuminated by a light source is called radiation and is represented by $L$. The camera sensor receives the radiation and creates an image according to the irradiance $I_x$, which is received at the position $x$. Since the lens barrel shades the light, the scene radiation falls along the image boundaries producing a vignetting effect, so the irradiance $I(x)$ can be obtained by multiplying the scene radiation $L$ by the vignetting factor $V(x)$. Each frame of the image taken by the auto-exposure camera will produce a corresponding exposure time $t$, and the cumulative irradiance can be obtained by integrating the irradiance within the exposure time $t$.

$$I_{acc}(x) = tI(x) = tV(x)L$$ (1)

The camera response function outputs the image intensity as a pixel value based on the cumulative irradiance $R \in [0, 255]$. For ease of representation, normalized $R \in [0, 255]$ to the unit interval $[0, 1]$. The output intensity of the entire image can be expressed as

$$O = f(tV(x)L)$$ (2)

The same scene $p \in P$ is observed by multiple frames and the photometric error can be expressed as

$$E = \sum_{p \in P} \sum_{i \in P} \omega^p \left\| O^p - f(eV(x^p) L^p) \right\|_b$$ (3)

Where $O^p$ denotes the luminosity of the image point in frame $I$, $\omega^p$ denotes the weight and $\| \|$ denotes the robust kernel function.

The camera response function is modelled using an empirical response model and the camera response function $f_G(x)$ can be expressed as

$$f_G(x) = f_0(x) + \sum_{k=1}^{n} c_k h_k(x)$$ (4)

$f_0(x)$ denotes the average response function, $h_k(x)$ is the basis function and $c_k$ denotes the weight.

Denoting the normalized radius of a pixel point $x$ relative to the image as $R(x)$, the vignetting factor $V(x)$ can be modelled as a sixth order polynomial.

$$V(x) = 1 + v_1 R(x) + v_2 R(x)^2 + v_3 R(x)^3 + v_4 R(x)^4 + v_5 R(x)^5 + v_6 R(x)^6$$ (5)

In order to accurately calculate the camera response function and vignetting factor, multiple frames are required for calibration. Since the exposure time can be calculated in real time, and the camera response function and vignetting factor need to be calibrated in multiple frames, the exposure time can be decoupled from other parameters. The photometric error of an image can be expressed as

$$E = \sum_{i=1}^{M} \sum_{p \in P} \omega^p \left( \frac{f^{-1}(O^p)}{V(x^p)} - e_i L^p \right)^2$$ (6)

The flow of the real-time photometric calibration is shown in Figure 2. Each time a new frame arrives, the exposure time is optimized according to the current value of the vignetting factor and response function, and the vignetting factor and response function are nonlinearly optimized and updated after accumulating multiple frames. To increase the efficiency of the system, only the key frames are calibrated in real time.
3.2 Line feature extraction and tracking

Because the LSD algorithm detects many short-length line features in the image, it is difficult to estimate the pose and occupy more computing resources. Therefore, the short line features with insufficient length are filtered out to improve the efficiency of line feature extraction. The minimum length of the line segments selected is

$$l_m = \mu \cdot \min\left(W_i, H_i\right)$$

where \(l_m\) denotes the minimum length of the line segment, and \(W_i, H_i\) the width and height of the image respectively. The weight is set to 0.1 to ensure the quality of the extracted line features while improving the extraction speed.

The LSD algorithm suffers from the problem of single lines breaking into multiple lines and lines at image boundaries being excised, and these lines cannot be successfully matched by LBD, resulting in degraded line feature tracking performance. In this paper, the line feature tracking method based on optical flow is used to supplement the matching results, thereby improving the performance of line feature tracking. The line feature tracking flow chart is shown in Figure 3.

3.3 Spatial line description

For the convenience of presentation, define the world coordinate system \(w\), the IMU coordinate system \(b\) and the camera coordinate system \(c\). Given a straight line in space \(L_w \in w\) it can be described by means of Plücker coordinates as follows.

$$L_w = \begin{bmatrix} n_w \\ d_w \end{bmatrix} \in R$$

\(n_w, d_w\) respectively represent the normal vector and direction vector in \(w\).
If \( L_w \) observed at two separate points in \( w \), the two planes \( L_w^1 \), \( L_w^2 \) can be determined, and the Plücker matrix can be identified as

\[
\begin{bmatrix}
[d]_n \\ -n^T \\
0
\end{bmatrix} = \pi_i \pi_j^T - \pi_j \pi_i^T \in \mathbb{R}^{4 \times 4}
\]

(9)

where \( [ \ ] \) denotes the partial symmetry matrix.

The Plücker coordinates are over-parametrized, therefore, an orthogonal vector with four degrees of freedom is used to orthogonally represent it.

\[
U = R(\theta) = \begin{bmatrix}
\frac{n}{\|n\|} \\
\frac{d}{\|d\|} \\
\frac{n \times d}{\|n \times d\|}
\end{bmatrix}, \quad W = \begin{bmatrix}
\cos(\phi) & -\sin(\phi) \\
\sin(\phi) & \cos(\phi)
\end{bmatrix} = \frac{1}{\sqrt{\|n\|^2 + \|d\|^2}} \begin{bmatrix}
\|n\| & -\|d\|
\end{bmatrix}
\]

(10)

The transformation of the Plücker coordinates of a straight line to an orthogonal vector \((U, W)\) is related as follows.

\[
L_w = \begin{bmatrix} w_1 u_1^T, w_2 u_2^T \end{bmatrix}
\]

(11)

Where \( \theta \) denotes a three-dimensional vector, \( \phi \) denotes a one-dimensional vector, and a four-degree-of-freedom orthogonal vector \( O \) can be expressed as

\[
O = (\theta, \phi)
\]

3.4 Line residual model and sliding window optimization

The transformation matrix of the line feature from \( w \) to \( c \) is

\[
T_{cw} = \begin{bmatrix}
R_{cw} & [t_{cw}] \times R_{cw} \\
0 & R_{cw}
\end{bmatrix}
\]

(13)

\( R_{cw} \) represents rotation and \( t_{cw} \) translation, the transformation matrix allows the linear coordinates \( L_w \) into \( L_c \).

\[
L_c = T_{cw} L_w = \begin{bmatrix}
R_{cw} & [t_{cw}] \times R_{cw} \\
0 & R_{cw}
\end{bmatrix} \begin{bmatrix}
[n_w] \\
[d_w]
\end{bmatrix}
\]

(14)

Transforming \( L_c \) to the pixel plane yields the projection line \( l \).

\[
l = K_L n_c
\]

(15)

\( K_L \) denotes the line projection matrix. According to the distance from the midpoint of the projection line to the straight line, the residual model of the reprojected line can be constructed as

\[
r_L(\bar{Z}_L, X) = d(m, l) = \frac{m^T l}{\sqrt{l_1^2 + l_2^2}}
\]

(16)

\( m \) represents the homogeneous coordinates of the midpoint of the line. Once the line feature residual model is obtained, it is added to the sliding window to be optimized with the other variables, and the complete state variables in the sliding window can be defined as

\[
X = [x_0, \ldots, x_n, \hat{\lambda}_0, \ldots, \hat{\lambda}_n, O_1, \ldots, O_n, \bar{Z}_L], \quad X_k = [p_{k'}, q_{k'}, v_{k'}, b_r, b_g]
\]

(17)

\( n_k, n_p, n_t \) respectively indicate the sequence numbers of key frames, spatial points and spatial lines in the sliding window. \( \hat{\lambda} \) represents the inverse depth information of the feature point and the first keyframe in which it is observed, and \( O \) represents the four-parameter orthogonal representation of the
spatial line, \( X_k \) contains the position, speed and direction of the IMU in \( w \) at time \( k \). According to the state variables, the objective function of the sliding window can be defined as

\[
X = \text{argmin} \left( I_p + e_i + e_p + e_t \right)
\]  \hspace{1cm} (18)

\( I_p \) denotes the marginalized a priori residuals, \( e_i \) denotes the IMU pre-integrated residuals, \( e_p \) denotes the point reprojection residuals, \( e_t \) represents the line feature residuals and can be defined as

\[
e_t = \sum_{(i,j) \in L} \rho \left( \left\| Z_{L_j}^i - X \right\|^2 / \sum_{l_j} \right)
\]  \hspace{1cm} (19)

\( \rho \) denotes the robust kernel function to remove the effect of the exclusion point.

### 3.5 Loopback detection

VINS-Fusion uses a DBOW2[17] bag-of-words model based on point features for feature retrieval in loopback detection. On this basis, a dictionary based on line features is added, which consists of LBD descriptors and word trees are obtained by k-means++ clustering. The flow of loopback detection is shown in Figure 4.

The system compares each frame of the input with the previous keyframe for parallax to determine if a loopback candidate is found. Whenever a loopback candidate is found, a similarity score is obtained by matching the BRIEF descriptor and LBD descriptor of the frame with the bag of words, and the total similarity score \( s_t \) can be expressed as

\[
s_t = \mu \left( \frac{n_p}{n_p + n_l} \right) s_p + (1 - \mu) \left( \frac{n_l}{n_p + n_l} \right) s_l
\]  \hspace{1cm} (20)

\( s_t \) is obtained by weighted summation of point and line features, \( \mu \) represents the weight of point features, \( n_p \) and \( n_l \) respectively represent the number of point features and line features, \( s_p \) and \( s_l \) respectively represent the score of point and line features. When the score exceeds a given threshold and passes the time and space consistency test, the frame is considered to be a loopback frame and the loopback constraint is obtained by optimizing the frame in four degrees of freedom \( e_o \).
\[ e_o = \sum_{(i,j) \in L} \left( \rho \| r_{i,j} \|^2 \right) \]  

(21)

\( r_{i,j} \) denotes the marginalized residual between the two frames. Add \( e_o \) to the back-end objective function for nonlinear optimization. The objective function in the sliding window can be updated to

\[ X = \text{argmin} \left( I_p + e_i + e_p + e_l + e_o \right) \]  

(22)

We select the New College dataset to perform a comparative test to evaluate the performance of the loop detection algorithm. The accuracy-recall curve was plotted as shown in Figure 5.

Incorrect loopback detection can lead to failure of the bit pose estimation, so the algorithm is more concerned with loopback recall at 100% accuracy. As can be seen in Figure 5, the loopback recall with a combination of point and line features improves by 10% over the loopback recall using only point features, while maintaining 100% accuracy.

4. Experimental analysis

This chapter conducts experiments on the method proposed above and shows the experimental results of PLVINS-Fusion. The performance of PLVINS-Fusion in terms of photometric calibration and localization accuracy is evaluated on the publicly available benchmark datasets TUM Mono and EuRoc[18] respectively, and the results are compared and analyzed. The algorithm experimental platform was implemented on a ROS system based on Ubuntu 18.04, and all experiments were performed on intel Core i7-10710U CPU@1.1.GHZ.

4.1 Photometric calibration experiments

We perform real-time photometric calibration on the TUM Mono dataset sequence 48, and compare the obtained photometric parameters with the true values provided by the dataset. The result of photometric parameter calibration is shown in the figure 6.
As shown in figure 6, the estimated values of camera response function, vignetting factor and exposure time are very close to the true value, which proves the effectiveness of the photometric calibration algorithm.

4.2 EuRoc dataset experiments
The performance of line feature tracking was evaluated by counting the number of successfully tracked line features in a sliding window of a certain number of frames, with lines that were tracked in at least five key frames being considered successfully tracked. Several sequences from the EuRoc dataset were selected for the line feature tracking count experiment. A comparison of the number of line features tracked is shown in Table 1.

| Sequence      | LSD+LBD | Proposed |
|---------------|---------|----------|
| MH_04_difficult | 22      | 29       |
| V1_02_medium   | 18      | 25       |
| V2_03_difficult| 33      | 51       |

It can be seen from Table 1 that in multiple sequences with rich line features, the improved line tracking algorithm based on optical flow can track more line features, and the overall tracking performance is about 40% higher than that of the traditional algorithm.
To compare trajectories before and after the improvement of the visual inertial odometry calculation method, evo was used as the evaluation tool, and the root mean square error (RMSE) of the trajectory was used as the criterion for evaluation. Table 2 shows a comparison of the accuracy of the algorithm for multiple sequences of the EuRoc dataset. w/o loop means loopback detection is turned off, w/ means loopback detection is turned on, and the data with smaller errors are bolded.

| Sequence           | w/o loop | w/ loop |
|--------------------|----------|---------|
|                    | VINS-    | PLVIO   | PLVINS- | VINS-    | PLVINS- |
|                    | Fusion   |         | Fusion  | Fusion   | Fusion  |
| MH_04_difficult    | 0.381    | 0.267   | 0.252   | 0.322    | 0.167   |
| MH_05_difficult    | 0.310    | 0.322   | 0.312   | 0.297    | 0.237   |
| V1_02_medium       | 0.112    | 0.130   | 0.123   | 0.104    | 0.093   |
| V1_03_difficult    | 0.197    | 0.147   | 0.141   | 0.188    | 0.122   |
| V2_02_medium       | 0.184    | 0.124   | 0.116   | 0.159    | 0.092   |
| V2_03_difficult    | 0.298    | 0.230   | 0.184   | 0.277    | 0.148   |
| Mean               | 0.242    | 0.203   | 0.190   | 0.224    | 0.146   |

The analysis of the data in Table 2 shows that compared to the VINS-Fusion algorithm, the algorithm incorporating line feature information reduces the trajectory error by an average of 22% with loopback turned off, and only performs slightly weaker than the original algorithm in the MH_05 and V1_02 sequences, and the trajectory accuracy is better than the original algorithm in all sequences with loopback turned on, with an average reduction of 40% in trajectory error, PLVIO does not include loopback detection and is only compared in the case of no loopback. From Table 1, we can see that the trajectory accuracy of PLVINS-Fusion is higher than that of PLVIO in all sequences, effectively improving the accuracy of the overall running trajectory. The reason for this phenomenon is that there are many low-texture and luminosity change scenes in this sequence, and the improved algorithm can effectively detect loopback in these scenes, thus further reducing the trajectory error.

A comparison of the output trajectory of the three algorithms with the true value of the trajectory under the MH_04 sequence is shown in Figure 7(a). Figure 7(b) compares the absolute position error versus time for the three algorithms. Figure 7(c) compares the error data for the entire process.
A comparison of the data from Figure 7 shows that the improved algorithm outputs smaller trajectory drifts in the MH_04 sequence than the original algorithm, closer to the true value of the trajectory, and an overall smaller absolute position error distribution. The absolute position error is significantly smaller than that of VINS-Fusion and PLVIO in the time stamps of 180 to 210, and the root mean square error as well as the maximum value are smaller than that of PLVIO and VINS-Fusion, which effectively improves the overall algorithm performance.

4.3 Analysis of algorithm time consumption

Whether an algorithm can run in real time is an important measure of its effectiveness. Table 3 shows the average computation time per frame for the different algorithms in the EuRoc dataset. The six main modules compared are photometric calibration, point feature extraction and tracking, line feature extraction, line feature tracking, sliding window optimization and loopback detection.

| Modules                      | Times (ms) |
|------------------------------|------------|
|                              | VINS-Fusion| PLVIO     | PLVINS-Fusion |
| Photometric Calibration      | 20         | ×          | ×             |
| Point Detection and Tracking | 15         | 15         | 15            |
| Line Detection               | ×          | 70         | 20 (Modified LSD) |
| Line Tracking                | ×          | 18         | 12            |
| Local VIO                    | 42         | 46         | 46            |
| Loop Closure                 | 200        | ×          | 240           |
As shown in Table 3, the improved algorithm adds a photometric calibration module and a line feature extraction and tracking module to the front end, and a line feature residual and point line feature loopback module to the back end, which increases the overall algorithm time consumption compared to the algorithm using only point features. Compared to the PL-VIO algorithm, which cannot run in real time, the proposed algorithm significantly reduces the time required for line feature extraction due to the improved LSD line feature extraction algorithm, and the improved line feature tracking method effectively improves the overall tracking performance.

5. Conclusion
Aiming at the problems of VINS-Fusion in low-texture and large luminosity scenes, such as decreased positioning accuracy and loopback error detection, this paper proposes a binocular visual inertial mileage calculation scheme PLVINS-Fusion that integrates point-line feature measurement information. The robustness of the algorithm to luminosity transformed scenes is improved by performing real-time photometric calibration of the image. Propose a loop detection method that can detect point and line features at the same time to improve the recall rate of loop detection. The comparative evaluation results of the TUM dataset and the EuRoc dataset show that PLVINS-Fusion meets the real-time requirements of the system, can obtain higher localization accuracy and better robustness than VINS-Fusion, and outperforms the PLVIO algorithm of the same type in overall performance.

References
[1] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM),” IEEE Robotics & Automation Magazine, 2006, 13(2): 99–116.
[2] Tribou, Michael, J. Multi-camera parallel tracking and mapping with non-overlapping fields of view[J]. The International Journal of Robotics Research, 2015, 34(12): 1480-1500.
[3] Handa A, Whelan T, McDonald J, Davison A J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM[C]/IEEE International Conference on Robotics and Automation, USA:IEEE, 2014:1524-1531.
[4] Leutenegger S, Lynen S, Bosse M, et al. Keyframe-based visual-inertial odometry using nonlinear optimization[J]. The International Journal of Robotics Research, 2015, 34(3):314-334.
[5] Sun K, et al. Robust stereo visual inertial odometry for fast autonomous flight[J]. IEEE Robot. Autom. Lett., 2018, 3(2): 965–972.
[6] Bloesch M, Burri M, Omari S, Hutter M, Siegwart R. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback[J]. Int. J. Robot. Res, 2017, 36(10):1053–1072.
[7] Geneva P, Eckenhoff K, Lee W, Yang Y, Huang G. Openvins: A research platform for visual-inertial estimation[C]/IEEE International Conference on Robotics and Automation. USA:IEEE, 2020:4666–4672.
[8] Engel J, Schöps T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM[C]/13th European Conference on Computer Vision, Berlin, Germany: Springer, 2014:834-849.
[9] Campos, Carlos, et al. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and MultiMap SLAM[J]. IEEE Transactions on Robotics, 2021.
[10] Bergmann P, Wang R, Cremers D. Online photometric calibration of auto exposure video for real-time visual odometry and SLAM[J]. IEEE Robot. Automat. Lett, 2018, 3(2):627-634.
[11] Zhang G, Lee J H, Lim J, Suh I H. Building a 3-D Line-Based Map Using Stereo SLAM[J]. IEEE Transactions on Robotics, 2015, 31(6):1364–1377.
[12] Gomez-Ojeda R, Moreno FA, Zuniga-Noël D, Scaramuzza D, Gonzalez-Jimenez J. PL-SLAM: a stereo SLAM system through the combination of points and line segments[J]. IEEE Transactions on Robotics, 2019, 35(3):734–746.
[13] Mur-Artal, Raúl, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System.” IEEE Transactions on Robotics 31.5(2015):1147-1163.
[14] He Y J, et al. PL-VIO: Tightly-coupled monocular visual inertial odometry using point and line
[13] Features [J]. IEEE Sensors, 2018, 18(4): 1159.

[15] Qin T, Li P L, Shen S J. VINS-Mono: A robust and versatile monocular visual-inertial state estimator [J]. IEEE Transactions on Robotics, 2018, 34(4): 1004–1020.

[16] Engel J, Koltun V, Cremers D. Direct sparse odometry [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(3): 611–625.

[17] Cummins M, Newman P. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance [J]. International Journal of Robotics Research, 2008, 27(6): 647-665.

[18] Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, Achtelik M W, Siegwart R. The EuRoC micro aerial vehicle datasets [J]. IEEE The International Journal of Robotics Research, 2016, 35(10): 1157–1163.