





# A Point-Line VIO System With Novel Feature Hybrids and With Novel Line Predicting-Matching

Hao Wei , Fulin Tang , Zewen Xu, Chaofan Zhang , and Yihong Wu 

**Abstract**—Weak texture and motion blur are always challenging problems for visual-inertial odometry (VIO) systems. To improve accuracy of VIO systems in the challenging scenes, we propose a point-line-based VIO system with novel feature hybrids and with novel predicting-matching for long line track. Point-line features with shorter tracks are categorized into “MSCKF” features and with longer tracks into “SLAM” features. Especially, “SLAM” lines are added into the state vector to improve accuracy of the proposed system. Besides, to ensure the reliability and stability of detection and tracking of line features, we also propose a new “Predicting-Matching” line segment tracking method to increase the track lengths of line segments. Experimental results show that the proposed method outperforms the state-of-the-art methods of VINS-Mono [1], PL-VINS [2] and OpenVINS [3] on both a public dataset and a collected dataset in terms of accuracy. The collected dataset is full of extremely weak textures and motion blurs. On this dataset, the proposed method also obtains better accuracy than ORB-SLAM3 [4].

**Index Terms**—Visual-Inertial SLAM, SLAM.

## I. INTRODUCTION

OVER the years, visual simultaneous localization and mapping (V-SLAM) or visual-inertial odometry (VIO) systems have been widely used for a range of applications, such as robot navigation, autonomous driving, and virtual reality [5], [6]. Most existing approaches rely on point features, which can perform well in textured scenes. In some challenging conditions, such as weak texture scenes and motion blur situations, the performance of point-based SLAM/VIO systems will degenerate or even collapse. At present, the combination of point and line segment features is an effective method to address those problems,

which has been investigated in many studies [2], [7]–[9]. However, the accuracy of the existing point-line-based SLAM/VIO systems is still insufficient in challenging conditions.

On the one hand, for VIO systems, feature detection and tracking are important for accuracy. The current detection and tracking of line features are still unstable [10]. In particular, in the weak texture scenes and motion blur situations, line extraction and tracking are easy to break and are with very short frames, which causes great decreases of the accuracy of VIO systems. To address this problem, we propose a novel “Predicting-Matching” line segment tracking method. For matching the failing lines, we directly calculate predicting lines and use the predicting lines to match with the detected lines. With the proposed method, we can obtain more line features with long tracks, which will improve accuracy of the proposed VIO system indirectly.

On the other hand, the current line features are not utilized in a SLAM/VIO system fully. To make full use of the information from line features, we propose a novel hybrid point-line-based VIO framework. In [11] and [3], it has been proved that the hybrid use of point features could improve the accuracy of VIO systems. Inspired by this, we combine point and line segment features in a novel hybrid way. The hybrid has three meanings: (1) hybrid use of point and line segment features, (2) hybrid use of “MSCKF” points and “SLAM” points, (3) hybrid use of “MSCKF” lines and “SLAM” lines. For the classical MSCKF method, there are only “MSCKF” features. In the proposed system, features with short tracks will be processed as “MSCKF” features, and features with long tracks will be processed as “SLAM” features. “SLAM” features are added into the state vector and updated iteratively to obtain accurate 3D positions. Afterward, the updated “SLAM” features are utilized to correct camera poses further. Therefore, the proposed method in this hybrid way can improve the accuracy.

The main contributions are summarized as follows:

- 1) A novel hybrid point-line-based VIO framework is proposed. For the first time, point and line segment features are both utilized in a hybrid way. We categorize “MSCKF” features as with shorter tracks and “SLAM” features as with longer tracks. They are used differently.
- 2) A “Predicting-Matching” line segment tracking method is proposed, which will significantly increase track lengths of line segments and improve accuracy of the proposed VIO system.
- 3) Experimental results validate the effectiveness of our methods. In particular, our method outperforms the state-of-the-art methods of VINS-Mono [1], PL-VINS [2] and

Manuscript received April 28, 2021; accepted September 8, 2021. Date of publication September 21, 2021; date of current version October 5, 2021. This letter was recommended for publication by Associate Editor J. Stueckler and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants 61836015 and 62002359, and in part by the Beijing Advanced Discipline Fund under Grant 115200S001. (Corresponding authors: Fulin Tang; Chaofan Zhang; Yihong Wu.)

Hao Wei, Zewen Xu, and Yihong Wu are with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100864, China (e-mail: weihao2019@ia.ac.cn; xuzewen2020@ia.ac.cn; yihong.wu@ia.ac.cn).

Fulin Tang is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100864, China (e-mail: fulin.tang@nlpr.ia.ac.cn).

Chaofan Zhang is with the Anhui Institute of Optics and Fine Mechanics, Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, P. R. China (e-mail: zcfan@aiofm.ac.cn).

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2021.3113987>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3113987

OpenVINS [3] on a public dataset and a collected dataset in term of accuracy. And also the proposed method obtains higher accuracy than ORB-SLAM3 [4] in weak texture and motion blur situations.

The rest of the letter is organized as follows. We discuss the related work in Section II. The details of the proposed hybrid point-line-based VIO system are presented in Section III. In Section IV, we introduce the proposed “Predicting-Matching” line tracking method. And in Section V, we provide experimental results. Conclusions and future work are given in Section VI.

## II. RELATED WORK

In this section, we review point-line-based SLAM/VIO systems, and please refer to [10] for a review of line segment tracking algorithms. There are two possible formulations of the estimator for SLAM/VIO which can be broadly categorized into the filtering-based methods and optimization-based methods [12].

**Optimization-based System:** Optimization-based approaches use bundle adjustment or graph to directly solve a nonlinear least-square problem over a set of measurements. Ruben *et al.* [9] used line segments to improve the robustness of SLAM systems in the scenes with weak-texture and illumination changes. In this work, 3D endpoint-pair parametrization is used to represent 3D line features. Zhang *et al.* [13] developed a stereo graph-SLAM system to build a 3D line-based map. Zuo *et al.* [14] presented a robust and efficient visual SLAM system that utilized point and line features. They [2], [7] leveraged line features to improve localization accuracy, which is based on VINS-Mono [1]. In [2], [7], [13], [14], the Plücker representation is used for 3D line initialization and projection, and orthonormal representation is used for optimization. Typically, optimization-based systems are more accurate, but with high computational cost [15], [16].

**Filtering-based System:** Compared with optimization-based systems, filtering-based methods are still popular due to their efficiency. In this letter, we focus on EKF-based algorithms. Thomas *et al.* [17] considered the internal constraints of Plücker representation during the update step of Kalman filter. Zhang *et al.* [18] presented a monocular method that uses vertical and floor line features to build a line-based partial 3D map. Zou *et al.* [19] proposed a visual-inertial SLAM system that uses line features with prior orientation, termed as struct lines. Zheng [15] presented a filtering-based stereo visual-inertial odometry method using both point and line features. Yang [20] designed a tightly-coupled monocular VIO system that leverages point and line features, and the designed system uses the closest point line representation. However, line landmarks were not included in the state vector.

This letter proposes an efficient filtering-based monocular VIO system using both point and line features. Different from [15] and [20], the line features with long tracks are added into the state vector, and the line features with short tracks are updated according to MSCKF [21] algorithm. Moreover, in order to improve the track lengths of line segments in weak

texture scenes, we propose a new “Predicting-Matching” line segment tracking method.

## III. THE PROPOSED HYBRID POINT-LINE-BASED VIO SYSTEM

In this letter, we propose a hybrid point-line-based VIO system, which is composed of image processing (Section III-A), hybrid point and line features (Section III-B), update MSCKF features (Section III-D), update SLAM features (Section III-C) and error-state Kalman filter (Section III-E), as shown in Fig. 1. When a new image comes into the proposed system, we use IMU sequences to propagate the state vector and covariance matrix. And then, we use feature measurements to update the system. We detect and track both point features and line segment features. For the first time, point and line features are both utilized in a hybrid way. They are divided into “MSCKF” features and “SLAM” features, based on their track lengths. We use the classical MSCKF algorithm to process the “MSCKF” features. Notably, “SLAM” features are added into state vectors and continuously iteratively optimized to obtain more accurate 3D positions. Lastly, both “MSCKF” and “SLAM” features can offer measurement Jacobian matrices to update the system state vector and covariance matrix.

The state vector of the proposed VIO system is comprised of the current IMU state  $\mathbf{x}_I^T$ ,  $c$  cloned IMU states in a sliding window  $\mathbf{x}_C^T$  denoting camera poses,  $m$  “SLAM” points  $\mathbf{x}_P^T$  and  $n$  “SLAM” lines  $\mathbf{x}_L^T$ . At time  $k$ , the state vector can be defined as:

$$\mathbf{x}_k = [\mathbf{x}_I^T \quad \mathbf{x}_C^T \quad \mathbf{x}_P^T \quad \mathbf{x}_L^T]^T. \quad (1)$$

The IMU state  $\mathbf{x}_I^T$  is described as [11]:

$$\mathbf{x}_I = \begin{bmatrix} I_k \bar{\mathbf{q}}^T & {}^G \mathbf{p}_{I_k}^T & {}^G \mathbf{v}_{I_k}^T & I_k \mathbf{b}_g^T & I_k \mathbf{b}_a^T \end{bmatrix}^T, \quad (2)$$

where  $I_k \bar{\mathbf{q}}^T$  is the unit quaternion representing the rotation from the global frame  $\{G\}$  to the IMU frame  $\{I\}$  at time  $k$ .  ${}^G \mathbf{p}_{I_k}^T$  and  ${}^G \mathbf{v}_{I_k}^T$  are velocity and position of the IMU with respect to  $\{G\}$ , and  $I_k \mathbf{b}_g^T$  and  $I_k \mathbf{b}_a^T$  are the gyroscope and accelerometer biases of IMU measurements, respectively.  $\mathbf{x}_C$  has the following form:

$$\mathbf{x}_C = \begin{bmatrix} I_{k-1} \bar{\mathbf{q}}^T & {}^G \mathbf{p}_{k-1}^T & \dots & I_{k-c} \bar{\mathbf{q}}^T & {}^G \mathbf{p}_{k-c}^T \end{bmatrix}^T. \quad (3)$$

The point features and line features in the state vector are given by:

$$\mathbf{x}_P = \begin{bmatrix} {}^G \mathbf{f}_1^T & {}^G \mathbf{f}_2^T & \dots & {}^G \mathbf{f}_{m-1}^T & {}^G \mathbf{f}_m^T \end{bmatrix}^T, \quad (4)$$

$$\mathbf{x}_L = \begin{bmatrix} {}^G \mathbf{l}_1^T & {}^G \mathbf{l}_2^T & \dots & {}^G \mathbf{l}_{n-1}^T & {}^G \mathbf{l}_n^T \end{bmatrix}^T, \quad (5)$$

where  $\mathbf{f}$  is a  $3 \times 1$  vector, and  $\mathbf{l}$  is a  $4 \times 1$  vector. There are four degrees of freedom for a spatial line. Therefore, we use orthonormal representation to model the line features for minimal parameterization.

### A. Image Processing

**Point Feature:** For an input image, we extract about 250 FAST corners on it. In particular, to make the extracted features

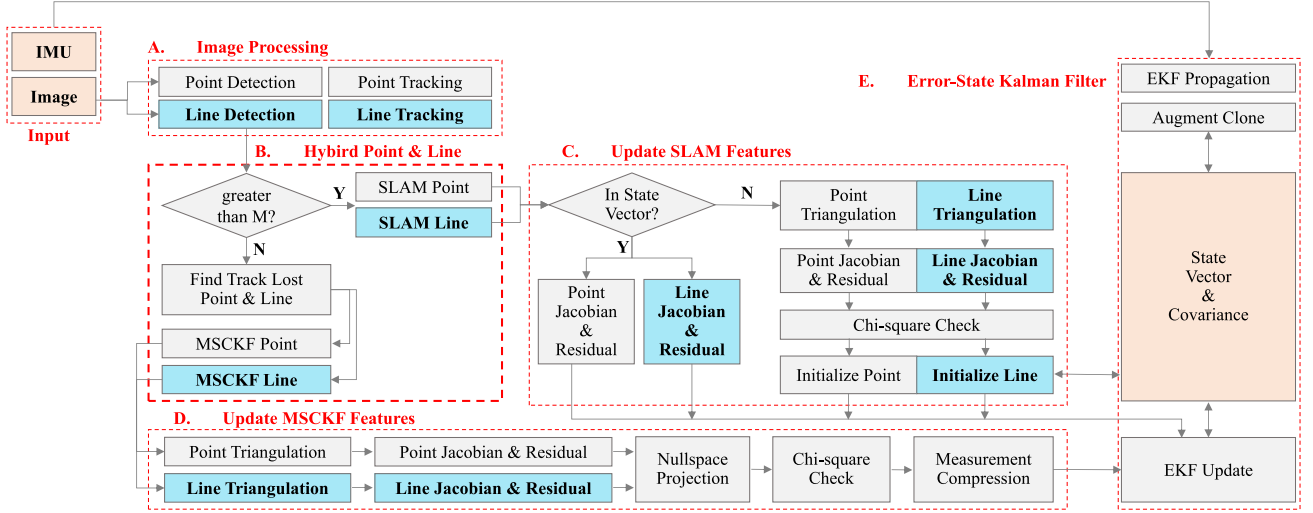


Fig. 1. A block diagram illustrates the pipeline of the proposed VIO system. The blue-filled boxes represent the processing steps of line features.

distribution evenly, we divide the input image into  $6 \times 4$  grids and then extract features in each grid. Afterward, we use the sparse KLT optical flow algorithm to track the extracted features. Last, the RANSAC algorithm is used to remove outliers.

**Line Feature:** The existing point-line-based VIO systems use LSD [22] to detect and LBD [23] to match line segments. However, both LSD [22] and LBD [23] are time-consuming. To reduce the computational complexity of the proposed system, we use FLD algorithm [24] to detect line segments. For line tracking, we propose a new “Predicting-Matching” line segment tracking method to improve the track lengths of line segments in weak texture scenes. Please refer to Section IV for more details.

### B. Hybrid Point and Line Features

The purpose of this part is to categorize point or line features as “MSCKF” features or “SLAM” features based on their track lengths. For the track results from Section III-A, we check their track lengths first. When the track length of a feature is more than a threshold, it will be added to the state vector, as “SLAM” features. We set the threshold to 15, which is the same as the number of camera poses in the state vector. We consider that “SLAM” features are stable features in the environment, and these features can provide strict geometric constraints on the camera poses. Therefore, “SLAM” features are added into state vectors and continuously iteratively optimized to obtain more accurate 3D positions. Besides, we observe that the number of “SLAM” lines is much less than “SLAM” points, which motivates us to propose a new segment tracking algorithm. The features whose track lengths are less than the threshold will be processed as “MSCKF” features. Those features will not be added to the state vector, and their measurements will only be used once.

### C. Update SLAM Features

For a “SLAM” feature, we check whether the feature has been added to the state vector first. If it is not added to the state

vector, we need to initialize it to the state vector. Otherwise, the new measurements of the feature will be used to update the state vector and covariance matrix directly. Since the error-state Kalman filter is used in the proposed system, point and line measurement models should be defined to obtain residuals. We focus on the line-based measurement model since the point-based measurement model has been widely studied [3], [11], [21].

**Line Feature Measurement Model:** For a line segment  ${}^G\mathbf{L}_{6 \times 1}$  in global frame  $\{G\}$ , its Plücker representation in camera frame  $\{C_j\}$  can be denoted as:

$${}^{C_j}\mathbf{L} = \begin{bmatrix} \mathbf{n}_{C_j} \\ \mathbf{d}_{C_j} \end{bmatrix} = {}^{C_j}_G T^G \mathbf{L} = \begin{bmatrix} {}^{C_j}_G \mathbf{R} & [G\mathbf{t}_{C_j}] \times {}^{C_j}_G \mathbf{R} \\ \mathbf{0} & {}^{C_j}_G \mathbf{R} \end{bmatrix} {}^G\mathbf{L}, \quad (6)$$

where  ${}^{C_j}_G \mathbf{R}$  and  $G\mathbf{t}_{C_j}$  are rotation matrix and translation vector from the global frame  $\{G\}$  to the camera frame  $\{C_j\}$ . And we use the following equation to project the  ${}^{C_j}\mathbf{L} = (\mathbf{n}_c^T, \mathbf{v}_c^T)^T$  to the image plane.

$$\mathbf{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = K \mathbf{n}_c = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix} \mathbf{n}_c, \quad (7)$$

where  $f_x, f_y, c_x, c_y$  are intrinsic parameters of the camera. And line segment residual  $e_l$  is built based on the point-to-line distance between the projected line  $\mathbf{l}$  and the tracked lines' endpoints  $\mathbf{x}_s$  and  $\mathbf{x}_e$ :

$$e_l = \begin{bmatrix} d(\mathbf{x}_s, \mathbf{l}) \\ d(\mathbf{x}_e, \mathbf{l}) \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{x}_s^T \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \\ \frac{\mathbf{x}_e^T \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix}, \quad (8)$$

where  $\mathbf{x}_s = [u_s, v_s, 1]^T$  and  $\mathbf{x}_e = [u_e, v_e, 1]^T$ , and  $u_s, v_s$  and  $u_e, v_e$  are pixel coordinates of start point and endpoint, respectively.

Now, we obtain the line reprojection error  $e_l$ . By linearizing the line feature measurement model for camera poses and for

“SLAM” line 3D positions, the measurement can be denoted as:

$$\mathbf{e}_l \simeq \mathbf{H}_x \tilde{\mathbf{x}} + \mathbf{H}_l^T \tilde{\mathbf{l}} + \mathbf{n}, \quad (9)$$

where  $\mathbf{H}_x$  and  $\mathbf{H}_l$  are Jacobians of the measurement to the state and the line position, respectively, and  $\mathbf{n}$  is a  $2 \times 1$  noise vector. Next, we will derive formulas to calculate  $\mathbf{H}_x$  and  $\mathbf{H}_l$ .

**Line Jacobian:** We use Plücker and orthonormal representation for line features. And we compute the Jacobians with respect to line orthonormal representation  $\Theta$  and camera pose  $T$  via the chain rules:

$$\mathbf{H}_l = \frac{\partial \mathbf{e}_l}{\partial \Theta} = \frac{\partial \mathbf{e}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_C} \frac{\partial \mathbf{L}_C}{\partial \mathbf{L}_W} \frac{\partial \mathbf{L}_W}{\partial \Theta} \quad (10)$$

$$\mathbf{H}_{x_c} = \frac{\partial \mathbf{e}_l}{\partial T} = \frac{\partial \mathbf{e}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_C} \frac{\partial \mathbf{L}_C}{\partial T}. \quad (11)$$

Next, we stack all line residuals and Jacobians as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{x_1} & \mathbf{H}_{l_1} \\ \dots & \dots \\ \mathbf{H}_{x_m} & \mathbf{H}_{l_m} \end{bmatrix}, \Delta = \begin{bmatrix} \mathbf{e}_{l_1} \\ \dots \\ \mathbf{e}_{l_n} \end{bmatrix}, \quad (12)$$

Lastly,  $\mathbf{H}$  and  $\Delta$  can be used to update the state vector and covariance matrix. Specially, we provide the update method of line features.

**Line Update:** For line features in the state vector, if new observations are obtained, the state vector needs to be updated gradually. We transform line segment from orthonormal to Plücker, and then we extract  $\mathbf{U}$  and  $\mathbf{W}$  matrices from Plücker representation ( $\mathbf{U} \in SO(3)$ ,  $\mathbf{W} \in SO(2)$ ) [13], and update  $\mathbf{U}$  and  $\mathbf{W}$  by Lie algebra as follows:

$$\mathbf{U}' = \mathbf{U} \exp([\delta \boldsymbol{\theta}]_{\times}) = \mathbf{U} (\mathbf{I} + [\delta \boldsymbol{\theta}]_{\times}), \quad (13)$$

$$\mathbf{W}' = \mathbf{W} \exp([\delta \boldsymbol{\phi}]_{\times}) = \mathbf{W} (\mathbf{I} + [\delta \boldsymbol{\phi}]_{\times}), \quad (14)$$

where  $\mathbf{U}'$  and  $\mathbf{W}'$  represent updated forms of  $\mathbf{U}$  and  $\mathbf{W}$ , respectively. After updating, the updated Plücker representation can be extracted from  $\mathbf{U}'$  and  $\mathbf{W}'$  and transformed into orthonormal representation to complete the updating of line segment feature in the state vector.

For “SLAM” lines not added to the state vector, we should calculate their 3D positions first, and then the above measurement model is used to calculate their Jacobians and residuals. Finally, these lines will be added to the state vector, and the corresponding covariance matrix will be set with the Jacobians  $\mathbf{H}_x$  and  $\mathbf{H}_l$ . In our implementation, we set the maximum number for the point features to 75 and line features to 25 in the state vector for efficiency. And the new features can not be added until the added features are removed from the state vector due to tracking failure.

#### D. Update MSCKF Features

For all “MSCKF” lines, we should calculate their 3D positions first. The line triangulation method is introduced as follows in detail.

**Line Triangulation:** Given a set of corresponding line measurements and camera poses, the 3D positions of the lines can be initialized from multiple views planes intersection. We use

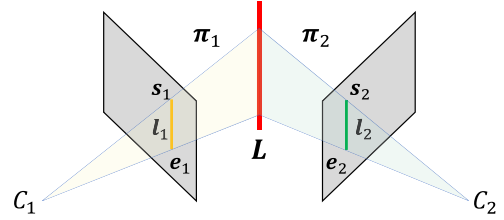


Fig. 2. Illustration of two-views line segment triangulation.

a line triangulation method which is based on  $n$ -views planes intersection [25].

One of the classical methods to triangulate line segment features is based on the two planes intersection [20]. In Fig. 2, the 3D line  $\mathbf{L}$  is projected to the  $C_1$  and  $C_2$  image planes as  $\mathbf{l}_1$  and  $\mathbf{l}_2$ , respectively. Let  $\mathbf{s}_1$ ,  $\mathbf{e}_1$  and  $\mathbf{s}_2$ ,  $\mathbf{e}_2$  be start points and endpoints of line segments  $\mathbf{l}_1$  and  $\mathbf{l}_2$  in the camera coordinate system, respectively. We use  $\mathbf{T}_j = \{ {}^G \mathbf{R}^{C_j}, {}^G \mathbf{t}_{C_j} \mid j \in 1, 2 \}$  to denote camera poses for  $C_1$  and  $C_2$ , where  ${}^G \mathbf{R}^{C_j}$  describe rotation matrix from global frame  $\{G\}$  to camera frame  $\{C_j\}$ , and  ${}^G \mathbf{t}_{C_j}$  is the optical center position of  $\{C_j\}$  in global frame  $\{G\}$ . Then the endpoints in global frame  $\{G\}$  can be represented as:

$$\begin{cases} {}^G \mathbf{s}_j = ({}^G \mathbf{R}^{C_j})^T {}^G \mathbf{s}_j + {}^G \mathbf{t}_{C_j} \\ {}^G \mathbf{e}_j = ({}^G \mathbf{R}^{C_j})^T {}^G \mathbf{e}_j + {}^G \mathbf{t}_{C_j}. \end{cases} \quad (15)$$

The plane formed by the camera center  ${}^G \mathbf{t}_{C_j}$  and the endpoints  $({}^G \mathbf{s}_j, {}^G \mathbf{e}_j)$  of the line segment  $\mathbf{l}_j$  can be represented as  $\pi_j$ :

$$\pi_j = \left[ ({}^G \mathbf{s}_j - {}^G \mathbf{t}_{C_j}) \times ({}^G \mathbf{e}_j - {}^G \mathbf{t}_{C_j}) \quad -{}^G \mathbf{t}_{C_j} ({}^G \mathbf{s}_j \times {}^G \mathbf{e}_j) \right]_{4 \times 1}^T, \quad (16)$$

and then the dual Plücker matrix  $L^*$  of 3D line  $\mathbf{L}$  can be computed as:

$$L^* = \pi_1 \pi_2^T - \pi_2 \pi_1^T = \begin{bmatrix} [\mathbf{v}]_{\times} & \mathbf{n} \\ -\mathbf{n}^T & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}. \quad (17)$$

Finally, we can directly extract Plücker coordinate  $(\mathbf{n}^T, \mathbf{v}^T)^T \in \mathbb{R}^{6 \times 1}$  of  $\mathbf{L}$ .

The camera motion will affect the performance of line triangulation. In [20], three kinds of degenerate motions that lead to line triangulation failure are analyzed. We use  $n$ -views line triangulation method to recognize the degenerate motion, and the lines in degenerate motions will not be used to triangulate, we will remove their measurements directly. For a 3D line  $\mathbf{L}$  which is projected to  $n$  image planes, we use  $\mathbf{M} \in \mathbb{R}^{n \times 4}$  to denote the plane matrix as:

$$\mathbf{M} = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_n \end{bmatrix}^T \in \mathbb{R}^{n \times 4}. \quad (18)$$

We carry out singular value decomposition for  $\mathbf{M}$ :

$$\mathbf{M} = \mathbf{U} \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^T, \quad (19)$$



**Algorithm 1: Predicting-Matching Line Tracking Method.****Require:**  $I_1, I_2, I_3, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, IMU_{1 \rightarrow 2}, IMU_{2 \rightarrow 3}, \mathbf{K}$ **Ensure:** Match result  $\mathbf{X}_{1 \rightarrow 2}, \mathbf{X}_{2 \rightarrow 3}$ 

- 1:  $\mathbf{X}_{1 \rightarrow 2} \leftarrow \text{FastMatch}(I_1, I_2, \mathbb{C}_1, \mathbb{C}_2, IMU_{1 \rightarrow 2}, \mathbf{K})$
- 2:  $\mathbb{F}_1 \leftarrow \text{MatchFail}(\mathbf{X}_{1 \rightarrow 2}, \mathbb{C}_1)$

**Step1: Predicting**

- 3:  $\mathbf{R} \leftarrow \text{Integration}(IMU_{1 \rightarrow 2}, \mathbf{K})$
- 4: **for**  $l_i \in \mathbb{F}_1$  **do**
- 5:    $\mathbb{P} \leftarrow \text{SamplePoints}(l_i)$
- 6:   **for**  $\mathbf{p}_j \in \mathbb{P}$  **do**
- 7:      $\mathbf{p}_j \leftarrow \text{KLT}(\mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{p}_j)$
- 8:      $\text{RemoveOutlier}(\mathbf{p}_j)$
- 9:     **if**  $\mathbf{p}_j \in \text{Inliers}$  **then**
- 10:        $\mathbb{P}' \leftarrow \mathbb{P}' \cup \mathbf{p}_j$
- 11:      $\mathbf{l}'_i, \mathbb{P}'_{inliers} \leftarrow \text{FitLine}(\mathbb{P}')$
- 12:    $\mathbb{L}_2 \leftarrow \mathbb{L}_2 \cup \mathbf{l}'_i$

**Step2: Matching**

- 13:  $\mathbb{M}_2 \leftarrow \text{OverlapRemove}(\mathbb{C}_2, \mathbb{L}_2)$
- 14:  $\mathbf{X}_{2 \rightarrow 3} \leftarrow \text{FastMatch}(I_2, I_3, \mathbb{M}_2, \mathbb{C}_3, IMU_{2 \rightarrow 3}, \mathbf{K})$
- 15: **return**  $\mathbf{X}_{1 \rightarrow 2}, \mathbf{X}_{2 \rightarrow 3}$

where  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$ , and  $\sigma_i (i = 1, 2, \dots, r)$  are all non-zero singular values.  $\sigma_1$  is the largest eigenvalue and  $\sigma_2$  is the second largest eigenvalue. Then, if  $\sigma_1, \sigma_2 \neq 0$ , we have two dominant planes  $\pi_1$  and  $\pi_2$ . The 3D line  $\mathbf{L}$  can be reconstructed as Equation (17). If  $\sigma_2 = 0$ , it indicates that the camera is in degenerate motion, which causes failures for line triangulation. And possible degenerate camera motions include camera motion along line direction, toward line direction and with pure rotation. After line triangulation, the method [13] is used to recover spatial 3D line endpoints for visualization.

For “MSCKF” lines, most operations are the same as “SLAM” lines. Differently, the Jacobian to line 3D positions need to be eliminated from (9), because “MSCKF” lines are not added in the state vector. We use left nullspace projection to eliminate  $\mathbf{H}_l$  in this letter.

**E. Error-State Kalman Filter**

The proposed VIO system uses IMU measurements to propagate the IMU state vector and covariance matrix. And the generic nonlinear IMU kinematics can be found in [20]. Lastly, the system performs an EKF update for state estimation.

**IV. PREDICTING-MATCHING LINE TRACKING**

In order to improve the track lengths of line segments in weak texture scenes and motion blur situations, we introduce a novel “Predicting-Matching” line segment tracking method in this section. And the flow of the proposed line tracking algorithm is shown in Algorithm 1. Since the proposed method can offer a longer track length, the result of line triangulation will be more

accurate. And the more precise 3D positions of line features also result in more accurate camera poses. The proposed line tracking method is partly based on our previous work [10], where we proposed a fast line matching algorithm. In this letter, we propose a novel “Predicting-Matching” strategy to increase the track lengths of line features.

As shown in Algorithm 1, the proposed line tracking algorithm contains predicting (Step1) and matching (Step2). For three consecutive images, denoted as  $I_1, I_2$  and  $I_3$ , we detect line segment features, denoted as  $\mathbb{C}_1, \mathbb{C}_2$  and  $\mathbb{C}_3$ , and we want to get line match results of  $I_1$  to  $I_2$  and  $I_2$  to  $I_3$ , denoted as  $\mathbf{X}_{1 \rightarrow 2}$  and  $\mathbf{X}_{2 \rightarrow 3}$  respectively. We use our previous fast matching algorithm [10] to get  $\mathbf{X}_{1 \rightarrow 2}$ .

Line detecting algorithms usually become unreliable under weak texture scenes or motion blur situations. For example, suppose a spatial line  $\mathbf{L}$  can be observed from  $I_1, I_2$ , and  $I_3$ , and the line does not be detected successfully in  $I_2$  due to serious motion blur, but the line can be detected successfully in  $I_1$  and  $I_3$ . In that case, the track length of the line is one. We use predicting lines to address this problem.

After getting  $\mathbf{X}_{1 \rightarrow 2}$ , we find lines that fail to match in  $\mathbb{C}_1$ , represented as  $\mathbb{F}_1$ . For each line in  $\mathbb{F}_1$ , we sample points  $\mathbb{P}$  on the line first and use an IMU-aided KLT algorithm to track the sampled points, and then use strict geometrical constraints to remove outliers. Next, we fit the remaining good track results  $\mathbb{P}'$  to get the predicting line  $\mathbf{l}'_i$  by a least-square method. In the matching step, we remove the overlap lines first. In particular, we prefer to retain the predicting lines since the predicting lines offer longer track lengths. Lastly, we use the fast matching algorithm again to get  $\mathbf{X}_{2 \rightarrow 3}$ . Especially, both predicting line and extracted lines in  $I_2$  are used to matching with lines in  $I_3$ . Compared with the frame-to-frame matching method, our method can get a longer track length, especially in the challenging environments.

**V. EXPERIMENTS**

In this section, the proposed system is tested on the EuRoC MAV dataset [26] and a collected dataset. We compare the proposed point-line-based VIO system with the state-of-the-art point-based systems (ORB-SLAM3 [4], OpenVINS [3], VINS-Mono [1]) and a point-line-based system (PL-VINS [2]). We disable the loop closing module in ORB-SLAM3 [4], VINS-Mono [1] and PL-VINS [2] to only test the odometry performance. Besides, for ORB-SLAM3 [4] and OpenVINS [3], we use their monocular-inertial configurations. All experiments have been run in an Intel Core i7-CPU, at 3.2 GHz, with 32 GB RAM. For all experimental results, we report the median of five times.

**A. Evaluation on EuRoC Dataset**

The EuRoC MAV dataset [26] consists of 11 sequences in three different environments, recorded by using a Micro Aerial Vehicle (MAV). Each sequence contains IMU measurements, synchronized stereo images, and accurate ground truth. In this letter, we only use images from the left camera and IMU measurements to evaluate the proposed VIO system. And we choose root mean squared error (RMSE) of the absolute trajectory error

TABLE I  
PERFORMANCE COMPARISON IN THE EUROC DATASET (RMSE ATE IN METER). THE LOOP CLOSING MODULE IS DISABLED FOR ALL SYSTEMS.  
AND ALL RESULTS ARE GAINED BY OUR COMPUTER WHILE KEEPING THEIR DEFAULT PARAMETER CONFIGURATION

Dataset	MH_01	MH_02	MH_03	MH_04	MH_05	V1_01	V1_02	V1_03	V2_01	V2_02	V2_03	Average
ORB-SLAM3 [4]	<b>0.019</b> <sup>1</sup>	<b>0.040</b> <sup>1</sup>	<b>0.038</b> <sup>1</sup>	<b>0.145</b> <sup>1</sup>	<b>0.076</b> <sup>1</sup>	<b>0.044</b> <sup>1</sup>	<b>0.015</b> <sup>1</sup>	<b>0.047</b> <sup>1</sup>	<b>0.045</b> <sup>1</sup>	<b>0.024</b> <sup>1</sup>	<b>0.078</b> <sup>1</sup>	<b>0.052</b> <sup>1</sup>
VINS-Mono [1]	0.155	0.178	0.195	0.348	0.302	0.089	0.111	0.188	0.086	0.158	0.278	0.190
PL-VINS [2]	0.157	0.170	0.227	0.303	0.282	0.070	0.123	0.180	<b>0.081</b> <sup>2</sup>	0.116	0.277	0.181
OpenVINS [3]	0.085	0.144	<b>0.095</b> <sup>2</sup>	0.273	0.275	0.063	0.087	<b>0.067</b> <sup>2</sup>	0.111	0.073	0.222	0.136
Ours w/o Line Hybrid	0.074	0.132	0.127	<b>0.224</b> <sup>2</sup>	0.285	0.068	0.091	0.072	0.105	0.065	0.187	0.130
Ours w/o Line Predict	0.084	0.123	0.110	0.241	<b>0.262</b> <sup>2</sup>	0.060	0.089	0.074	0.114	0.064	0.180	0.127
Ours	<b>0.071</b> <sup>2</sup>	<b>0.123</b> <sup>2</sup>	0.108	0.241	0.267	<b>0.057</b> <sup>2</sup>	<b>0.082</b> <sup>2</sup>	0.073	0.115	<b>0.064</b> <sup>2</sup>	<b>0.144</b> <sup>2</sup>	<b>0.122</b> <sup>2</sup>

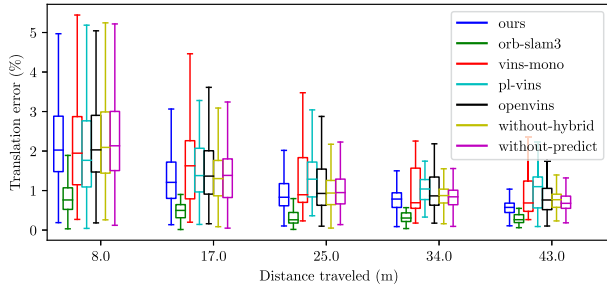


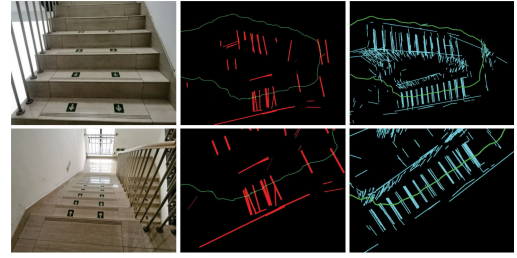
Fig. 3. Relative pose error(RPE) comparison in V2-03 sequence.

(ATE) [27] and relative pose error (RPE) as the evaluation metric. The ATE directly calculates the difference between the ground truth and the estimated camera pose, which reflects the overall performance of system. The RPE describes the pose difference of two frames during a fixed time interval. It reflects drift of the system. We compare accuracy of the proposed VIO system with the state-of-the-art methods, and the results of ATE and RPE are shown in Table I and Fig. 3 respectively. Obviously, ORB-SLAM3 [4] achieves more accurate results than others in all sequences due to local map refinement and global bundle adjustment. But the trajectories of ORB-SLAM3 [4] are not real-time, while other methods output real-time trajectories. This means that the history camera poses of ORB-SLAM3 [4] are optimized over time. Furthermore, the later experiments show that accuracy of ORB-SLAM3 decreases greatly in challenging scenes with weak texture and motion blur.

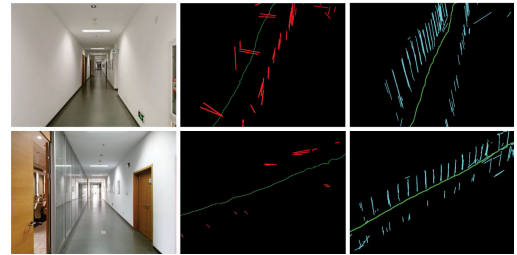
We can further observe that line segment features can improve the performance on most sequences. For example, PL-VINS [2] is more accurate than VINS-Mono [1], and the proposed method get better accuracy than OpenVINS [3]. In summary, our method using point and line features in a hybrid way achieves smaller errors on most sequences. Particularly, in the most challenging sequence V2\_03, our method obtains the best performance among VINS-Mono [1], PL-VINS [2] and OpenVINS [3] with a wide margin. What's more, when the hybrid line step is turned off, or the proposed "Predicting-Matching" line tracking method is not used, accuracy of the proposed system decreases significantly. It proves the effectiveness of the proposed system further.

### B. Evaluation on Collected Datasets

To further evaluate the proposed method under real-world scenarios, we record monocular images and IMU measurements



(a) Stairs Scene



(b) Weak-texture Scene

Fig. 4. Line maps build by PL-VINS [2] and Ours. The green lines show the camera trajectories. The left column images are examples in the scenes, the middle column images are the results from PL-VINS [2], and the right column images are the results from our method.

by holding a mobile phone (HUAWEI P30 Pro) in typical indoor environments, including staircases, corridors, and offices. The collected dataset contains four sequences in 2 different scenes. In each scene, we collect two sequences, named as Easy and Hard, based on different movement patterns. Both sequences have challenging scenes such as weak texture scenes and motion blur situations, as shown in the left column of Fig. 4. To evaluate the proposed VIO method quantitatively in the collected dataset, we enforce starting and ending points of each sequence to be at the same position. And we use a printed ArUco marker [28] to get camera trajectories at the beginning and the end for all sequences. For all sequences, about 15.6% of all images have ground-truth camera poses. As a result, the accumulated drift along the whole trajectory can be measured, which is considered as our evaluation metric, as [19] does.

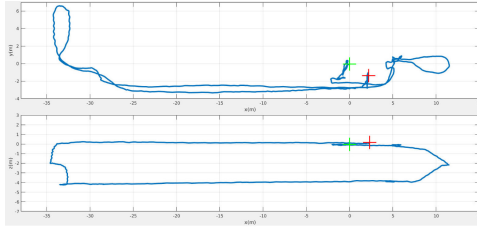
We compare the proposed system with the state-of-the-art methods, and the results are shown in Table II. The results show the superiority of our method in the weak-texture scenes and motion blur situations. Besides, we observe that the accuracy of optical flow-based methods (VINS-Mono [1], PL-VINS [2],

TABLE II  
PERFORMANCE COMPARISON IN THE COLLECTED DATASET (RMSE ATE IN METER)

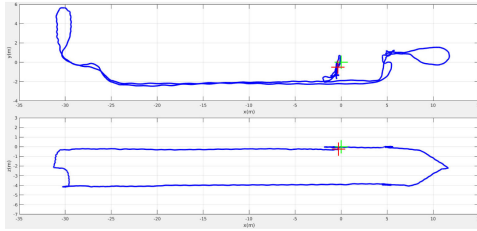
Dataset	ORB-SLAM3	VINS-Mono	PL-VINS	OpenVINS	Ours w/o Hybrid Line	Ours w/o Predict Line	Ours	Length(m)
Weak-texture Easy (W_E)	—	2.976	5.079	1.198	1.102	0.997	<b>0.979</b>	154
Weak-texture Hard (W_H)	—	1.690	1.784	1.037	0.917	0.888	<b>0.755</b>	147
Stairs Easy (S_E)	0.563	1.077	0.474	0.337	0.320	0.192	<b>0.108</b>	128
Stairs Hard (S_H)	0.862	0.803	1.332	0.501	0.516	0.250	<b>0.229</b>	126
Average	0.713*	1.637	2.167	0.768	0.714	0.582	<b>0.518</b>	138

TABLE III  
AVERAGE TRACK LENGTH OF LINE SEGMENTS ON THE EUROC DATASET

Dataset	MH_01	MH_02	MH_03	MH_04	MH_05	V1_01	V1_02	V1_03	V2_01	V2_02	V2_03	Average
[10]	3.1	2.9	3.0	3.1	3.1	2.9	2.8	2.2	3.1	2.7	2.2	2.8
The Proposed Method	6.5	5.4	5.9	7.2	6.5	7.9	5.2	4.3	8.0	5.1	4.0	<b>6.0</b>



(a) PL-VINS



(b) Ours

Fig. 5. Trajectories of PL-VINS [2] and Ours obtain on the Stairs Hard sequence. In this case, the camera travels on the second floor and then on the first floor. Both trajectories on  $xz$  and  $xy$  planes are shown. The proposed method outperforms the SOTA point-line-based method, PL-VINS [2].

OpenVINS [3] and Ours) is better than that of ORB-SLAM3 [4] in challenging environments. In Weak-texture Easy and Weak-texture Hard sequences, there are corridors with almost pure white walls. In these sequences, ORB-SLAM3 [4] can not extract and match enough stable feature points, which leads to corruption. In particular, our method uses line features and achieves the best accuracy. Compared with OpenVINS [3], the average RMSE ATE error of our method is reduced by 0.169 meters. ORB-SLAM3 [4] can run successfully in the stair scene, but obtains worse accuracy due to severe motion blur. In addition, when the hybrid line step is turned off, or the proposed “Predicting-Matching” line tracking method is not used in the collected dataset, accuracy of the proposed system decreases significantly.

Fig. 4 shows the line map obtained by the two point-line-based methods (PL-VINS [2] and Ours). We can observe that the line map built by PL-VINS [2] is relatively sparse and can not reveal the structural information. However, our method can obtain an accurate 3D line map in both stair scene and long straight

TABLE IV  
AVERAGE TRACK LENGTH OF LINE SEGMENTS ON THE COLLECTED DATASET

Dataset	W_E	W_H	S_E	S_H	Average
[10]	2.7	2.1	2.7	2.1	2.4
The Proposed Method	5.5	3.8	5.5	3.5	<b>4.6</b>

TABLE V  
EXECUTION TIME COMPARISON ALONG PL-VINS, OPENVINS AND OURS

Operation	Mean execution time (ms)		
	PL-VINS [2]	OpenVINS [3]	Ours
Point Detection & Tracking	15.0	3.0	3.0
Line Detection & Tracking	32.0	—	13.0
MSCKF Point Update	—	6.3	6.3
MSCKF Line Update	—	—	0.4
SLAM Point Update	—	4.5	4.5
SLAM Line Update	—	—	0.2
Others	46.0	7.7	7.9
Total	93.0	<b>21.5</b>	<b>35.3</b>

corridor scene. Fig. 6 and Fig. 5 show the trajectories on the Weak-texture Hard and Stairs Hard sequences, respectively. We can see that the trajectories of the proposed method are smoother and have less drift than others.

### C. Predicting-Matching Line Tracking Method

In this part, we compare the average track length of line features between the proposed method and the line tracking method in [10] on the EuRoC and the collected dataset. The results are shown in Table III and Table IV. We can see that the average line length of line segments is significantly increased when using the proposed “Predicting-Matching” line tracking method.

### D. Computation Time

Finally, we compare the computation time between the proposed method and PL-VINS. We run them on the EuRoC MH\_04 sequence, with  $752 \times 480$  image size. PL-VINS [2] is an optimization-based methods. OpenVINS [3] and our method are both filtering-based methods. From Table V, we can observe that the filtering-based methods are more efficient than the optimization-based method. The speed of our method is nearly



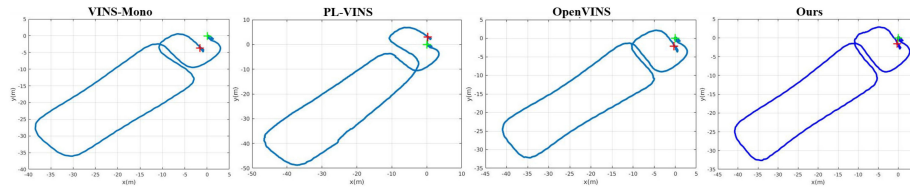


Fig. 6. Trajectories of Low-texture Hard sequence. In this case, the camera travels in corridors with weak texture and motion blur. The green '+' is the start point, and the red '+' is the endpoint. The proposed method (blue) outperforms others.

2.6 times faster than that of PL-VINS [2]. For line segment extraction and tracking module, our line extraction and tracking method takes about 13.0 ms to process per image, while PL-VINS [2] method takes 32 ms. In addition, it can be observed that estimating MSCKF and SLAM line segment features in the state vector for the proposed method only increases the time consumption by 0.6 ms.

## VI. CONCLUSION

In this letter, we propose a point-line-based VIO system with a novel feature hybrid way and a novel predicting-matching line segment tracking to deal with challenging scenes such as weak texture and motion blur situations more robustly. For the first time, point and line segment features are both utilized in a hybrid way. To improve accuracy of the proposed VIO system further, we propose a new "Predicting-Matching" line segment tracking method. With the tracking method, the track lengths of line segments in weak texture scenes are increased significantly, which improves the accuracy indirectly. Extensive experiments validate that the proposed method outperforms the state-of-the-art methods (VINS-Mono [1], PL-VINS [2] and OpenVINS [3]) on a public dataset and a collected dataset in term of accuracy. The collected dataset is full of weak texture and motion blur. Meanwhile, the speed of the proposed method is 2.6 times faster than that of the state-of-the-art point-line-based VIO system of PL-VINS. In the future, we will employ planes in SLAM systems to enhance accuracy and build meaningful geometrical maps.

## REFERENCES

- [1] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
- [2] Q. Fu, J. Wang, H. Yu, I. Ali, F. Guo, and H. Zhang, "Pl-vins: Real-time monocular visual-inertial slam with point and line," 2020, *arXiv:2009.07462*.
- [3] P. Geneva, K. Eickenhoff, W. Lee, Y. Yang, and G. Huang, "Open-vins: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4666–4672.
- [4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam," *IEEE Trans. Robot.*, pp. 1–17, 2021, *arXiv:2007.11898*.
- [5] Y. Wu, F. Tang, and H. Li, "Image-based camera localization: An overview," *Vis. Comput. Ind. Biomed., Art.*, vol. 1, no. 1, pp. 1–13, 2018.
- [6] F. Tang, Y. Wu, X. Hou, and H. Ling, "3D mapping and 6D pose computation for real time augmented reality on cylindrical objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2887–2899, Sep. 2020.
- [7] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "Pl-vio: Tightly-coupled monocular visual-inertial odometry using point and line features," *Sensors*, vol. 18, no. 4, 2018, Art. no. 1159, [Online]. Available: <https://www.mdpi.com/1424-8220/18/4/1159>.
- [8] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4503–4508.
- [9] R. Gomez-Ojeda, F.-A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "Pl-slam: A stereo slam system through the combination of points and line segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.
- [10] H. Wei, F. Tang, C. Zhang, and Y. Wu, "Highly efficient line segment tracking with an imu-klt prediction and a convex geometric distance minimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 3999–4005.
- [11] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Robotics: Sci. Syst.*, Berlin Germany, 2013, pp. 241–248.
- [12] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 9572–9582.
- [13] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-D line-based map using stereo slam," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, Dec. 2015.
- [14] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual slam with point and line features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1775–1782.
- [15] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C.-C. Chu, and H. Hu, "Trifo-vio: Robust and efficient stereo visual inertial odometry using points and lines," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3686–3693.
- [16] F. Tang, H. Li, and Y. Wu, "Fmd stereo slam: Fusing mvj and direct formulation towards accurate and fast stereo slam," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 133–139.
- [17] T. Lemaire and S. Lacroix, "Monocular-vision based slam using line segments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 2791–2796.
- [18] G. Zhang and I. H. Suh, "Building a partial 3D line-based map using a monocular slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1497–1502.
- [19] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, "Structvio: Visual-inertial odometry with structural regularity of man-made environments," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 999–1013, Aug. 2019.
- [20] Y. Yang, P. Geneva, K. Eickenhoff, and G. Huang, "Visual-inertial odometry with point and line features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2447–2454.
- [21] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3565–3572.
- [22] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, 2008.
- [23] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *J. Vis. Commun. Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.
- [24] J. H. Lee, S. Lee, G. Zhang, J. Lim, W. K. Chung, and I. H. Suh, "Outdoor place recognition in urban environments using straight lines," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 5550–5557.
- [25] S. J. Lee and S. S. Hwang, "Elaborate monocular point and line slam with robust initialization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1121–1129.
- [26] M. Burri *et al.*, "The euroc micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.
- [27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.
- [28] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image Vis. Comput.*, vol. 76, pp. 38–47, 2018.