DPLVO: Direct Point-Line Monocular Visual Odometry

Lipu Zhou[1], Shengze Wang[2] and Michael Kaess[3]

***Abstract*—In this paper, we present a direct visual odometry (VO) using points and lines. Direct methods generally choose pixels with sufficient gradients to minimize the photometric error for the status estimation. Pixels on lines are generally involved in this process. But the collinear constraint among these points are generally ignored, which may result in less accurate depth estimation. This paper introduces the collinear constraint into the state-of-the-art direct visual odometry DSO [1] to overcome this problem. The 3D lines, points and poses within a sliding window are jointly optimized. DSO implicitly establishes the data association for points among the keyframes within a sliding window by direct image alignment. This scheme is typically suitable for points, as points are generally only visible within a short time window. However, as lines are unbounded entities, they can be observed by a camera significantly longer than points. Thus, we seek to establish the long-term data association for lines among the keyframes. The 3D collinear points that are removed from the sliding window are served as collinear priors for the following windowed optimization. We prove that the prior collinear constraints of a 3D line can be compressed into six residuals in the optimization. This significantly reduces the computational complexity, and enables real-time performance for incorporating long 3D line segments into the windowed optimization. We present a new 3D line representation which reduces the four degrees of freedom (DoF) of a 3D line into two DoF by parameterizing the 3D line in the back-projection plane of its first 2D line observation. The experimental results show that our algorithm outperforms the state-of-the-art direct monocular VO algorithms.**

***Index Terms*—SLAM, Mapping, Localization**

I. Introduction

Simultaneous localization and mapping (SLAM) is a fundamental problem in robotics and computer vision community. Due to its importance, SLAM using various sensors has been extensively studied. Among them, monocular visual SLAM (VSLAM) and visual odometry (VO) have received tremendous attentions, as a monocular camera is widely available. Generally, VSLAM and VO algorithms can be classified into two categories: feature-based and direct methods. In the feature-based method, such as PTAM [2] and ORB-SLAM [3], the descriptors of salient features are extracted and matched along the video sequence. On the other hand, the direct approaches, such as LSD-SLAM [4] and DSO [1], leverage raw pixel intensities and minimize the photometric error. As the direct method can employ more information, it is generally more robust in poorly textured scenes [1]. Direct methods typically adopt pixels with sufficient gradients, which generally include pixels on lines. These pixels may have one dimensional freedom along the line during photometric tracking, which may result in less accurate depth estimation. We introduce collinear constraints into DSO [1] to improve the depth accuracy of the points on lines.

Fig. 1. An example of long-term line association. As lines are unbounded, they can generate long-term associations. This increases the correlation among poses, which benefits reducing drift, but may also significantly increase the runtime of the optimization if all the poses where a long 3D line segment is observed are jointly optimized. Thus it is important to design a scheme to balance the efficiency and accuracy. In addition, the line matching is also challenging. Since different parts of a line may be captured at different time, as demonstrated above, the feature-based matching scheme which is well-established for points is not suitable for lines.

Lines widely exist in man-made environments. They are good complements to points. However, it is not trivial to exploit lines into VSLAM or VO. A line is an infinite object. Thus it can exist in the field of view (FoV) of a camera much longer than a point, which adds more correlation among poses, as demonstrated in Fig. 1. Although this benefits reducing the drift, it may increase the runtime of the optimization as well. In feature-based methods, the general strategy to employ lines is to detect and match lines [5]–[14]. As the line detection algorithm, such as LSD [15], generally cannot ensure to detect the same line segment in each frame, this may cause the line matching unstable. The deep learning based line detector [16] has got significant progress in recent years. But these algorithms require a powerful GPU, which is generally not available for embedded systems where the computation resource is limited. Additionally, once a part of a line segment moves out or emerge into the camera's FOV, the line matching may also fail. Furthermore, the line detection and matching significantly increase the computational load [5]. On the other hand, direct methods minimize the photometric error. This strategy is suitable for points, but it is difficult to be extended to lines. Thus, the number of studies on extending the direct method with lines is relatively small. In the literature, lines are generally not involved in the optimization in direct methods [17], [18]. Moreover, it is considered infeasible to add geometry prior on points in the direct method for real-time applications [1].

This paper seeks to improve the accuracy of the state-of-the-art direct method DSO [1] by establishing long-term line association and introducing collinear constraints into the windowed optimization. The main contributions of the paper are:

- We introduce the collinear constraint into the photometric error of DSO [19]. We establish long-term data association among different 2D line observations of a 3D line. We fix the poses of the keyframes that are removed from the sliding window and the depths of the collinear points in these removed keyframes to yield the prior collinear constraints for the following windowed optimization. We prove that the prior collinear constraints of a line are equivalent to six residuals in the optimization, which is independent of the number of prior collinear constraints. This makes incorporating long 3D line segments into the

Manuscript received: February 24, 2021; Revised: May 31, 2021; Accepted: June 24, 2021.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work was partially done when the first two authors were with the Robot Perception Lab at Carnegie Mellon University.

[1]Lipu Zhou is with Magic Leap, 1376 Bordeaux Dr, Sunnyvale, CA 94089, USA lzhou@magicleap.com

[2]Shengze Wang is with Department of Computer Science, University of North Carolina, 3175 Brooks, Chapel Hill, NC 27599, USA shengzew@cs.unc.edu

[3]Michael Kaess is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA kaess@cmu.edu

Digital Object Identifier (DOI): see top of this page.

optimization and keeping the real-time performance feasible.
- We present a new singular-free 3D line representation with two degrees of freedom (DoF). It is known that a 3D line has four DoF. The key point of our approach is to represent a line in the back-projection plane of the first 2D line observation.

II. RELATED WORK

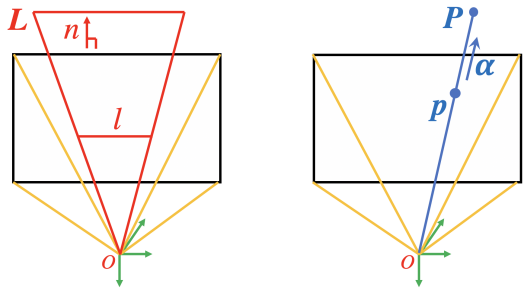
Points are the most common features used in VSLAM and VO algorithms [1]–[3], [19], [20]. On the other hand, Lines are abundant in man-made scenes. Thus as an important complement to points, lines have received extensive attention in recent years.

Feature-based method with lines The feature-based method using points alone is prone to fail in low textured scenes, where abundant lines generally exist. As a counterpart of points, it seems natural to extend the well established feature-based framework of points to lines [5]–[14]. In these algorithms, lines are matched along the sequence as done for points. The drawback of this scheme is that there is no guarantee that the same segment of a 3D line can be repetitively detected along the sequence. Since the descriptors may represent different parts of a 3D line, the line matching may be unstable. Furthermore, this scheme cannot establish long-term data association. As a line is an infinite object, the segment of a 3D line captured by a moving camera generally changes. For instance, a new part of a 3D line may gradually emerge into the camera's FoV, meanwhile an observed part may move out of the camera's FoV, as demonstrated in Fig. 1. The feature-based method focuses on matching the first detected 2D line segment in the following frames, thus it probably fails once the captured line segment changes.

Direct method with lines The direct method minimizes the photometric error, which is difficult to be extended for lines. Thus the number of studies on the direct method with lines is relatively small. As direct methods generally employ pixels with sufficiently high gradients, pixels on lines are generally selected in the photometric cost. In previous works [17], [18], the 3D lines are used to force the collinear 3D points to meet the collinear constraint. Specifically, the depths of the collinear points are estimated as done for the normal points. The 3D lines are fitted from the estimated 3D points. Then the 3D points are projected into the corresponding 3D lines. Gomez-Ojeda *et al.* [21] introduce lines into the semi-direct method SVO [22]. In [21], a line is represented by two points, and the depth filter introduced in [22] is adopted to estimate the depth of the endpoints of lines. In these methods, the 3D lines themselves are not jointly optimized with the keyframe poses and points. This may results in suboptimal results.

Structural Line The indoor scenario is generally structured, known as the Manhattan world model that has three perpendicular line directions. The lines with these directions are called the structural lines [23], [24]. Vanishing points estimated from the parallel structural lines can be used to identify these directions, which in turn benefit the estimation of the camera orientation [25]–[27]. Some works [23], [24] introduce the structural lines into the extend Kalman filter (EKF) framework. Recently, Zou *et al.* [24] adopt the Atlanta world model to generalize the Manhattan world model. Non-structural lines, which are also important for pose estimation in the low texture areas, are ignored in these works. Besides, the Manhattan assumption limits its applicable scenarios.

Line Representation Representing a 3D Line in the optimization is not trivial. It is known that a 3D line has four DoF [28]. A 3D line can be straightforwardly represented by two 3D points, which is adopted in some previous works [5], [7], [9]. This representation has six DoF. Solà *et al.* [29] present five 3D line representations. But none of them are minimal representation. The overparameterization may result in a rank-deficient Hessian matrix, which may lead to slow convergence and suboptimal results for the Levenberg-Marquardt (LM) algorithm [30]. In [31] and [32], two minimal representations for 3D lines depending on the representation of the rotation matrix are presented. As every minimal representation of the rotation matrix has singular cases, the two representations heritage these singularities. Furthermore, both representations are singular when a 3D line passes through the origin of the coordinate system. Kottas *et al.* [33] adopt a quaternion and a distance scalar to represent a 3D line. Yang and Huang [34] multiply the quaternion and the distance scalar together to generate a four DoF representation, named the closed point (CP) representation. Like [31] and [32], the CP representation is singular when a 3D line passes through the origin. In [23] and [24], they customize the representation for the structural line in the Manhattan world. Li *et al.* [13] propose a parameterization for co-planar lines. This paper presents a 2D singularity-free representation for 3D lines using the back-projection plane of the first 2D line observation.



(a) Back-projection plane of a 2D line

(b) Back-projection line of a 2D point

Fig. 2. The back-projections of a 2D line $\boldsymbol{l}$ and a point $\boldsymbol{p}$ on the image plane are a plane and a 3D line which pass through the origin of the camera coordinate system, respectively. $\boldsymbol{l}$ and $\boldsymbol{p}$ determine the normal $\boldsymbol{n}$ of the back-projection plane and the direction $\boldsymbol{\alpha}$ of the back-projection line, respectively. The 3D line $\boldsymbol{L}$ is on the back-projection plane of $\boldsymbol{l}$. The 3D point $\boldsymbol{P}$ is on the back-projection line of $\boldsymbol{p}$.

III. SYSTEM OVERVIEW

This work introduces lines into DSO [1]. Thus we extend the back end and font end of DSO for lines.

In the back end, we introduce a novel singular-free line parameterization which only has 2 DoF. We impose the collinear constraint on the depths of the 2D points sampled from a 2D line. To adapt the windowed optimization framework of DSO, we introduce the collinear prior for the collinear points that are removed from the sliding window. We prove that, no matter how many collinear priors there are, they can be compressed into six residuals, and can be calculated incrementally. This significantly reduces the computational complexity for long 3D line segments.

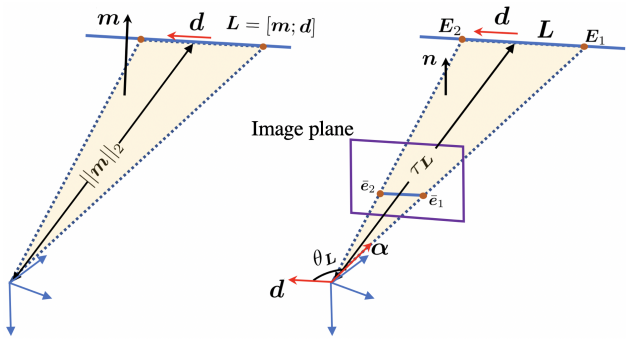
In the front end, we introduce the line management which consists of line detection, line initialization and line association. We adopt the LSD algorithm [15] to detect lines, and seek to combines lines with similar parameters. We only detect lines in the area that is not associated to an existing line. For each new 2D line segment, we sample some points from it according to the length of the line segment. The depth of each point is estimated as the ordinary points in DSO. These depths are then used to initialize the 3D line. To avoid the disadvantage of the descriptor-based line matching scheme mentioned above, we introduce a tracking-extension-redistribution method to establish the long-term line association among keyframes.

IV. NOTATIONS AND PRELIMINARIES

In this paper, we use boldfaced letters to represent vectors and matrices, and employ italic lowercase and italic uppercase to represent scalars and functions, respectively.

Camera Pose Let us denote the rotational and translational components of a camera pose transforming a point from the world coordinate system to the camera coordinate system as $\mathbf{R} \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$, respectively. The pose of a camera is then represented by $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \boldsymbol{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \in SE(3)$.

Projection and Back-projection In this paper, the projection from a 3D point to the image plane is denoted as $\Pi_{\mathbf{c}} : \mathbb{R}^3 \rightarrow \Omega$, where $\mathbf{c}$ represents the camera parameters. Accordingly, given a pixel



(a) Plücker Coordinates (b) Parameterize $\boldsymbol{L}$ by $\theta_{\boldsymbol{L}}$ and τ_L

Fig. 3. A schematic of Plücker Coordinates (a) and our new line parameterization (b). Given a 2D line, the 3D line is on its back-projection plane, as demonstrated in Fig. 2. Thus we can represent a 3D line by 2 parameters (*i.e.*, $\tau_{\boldsymbol{L}}$ and $\theta_{\boldsymbol{L}}$). The parameterization is introduced in section V-B1.

and its depth, the back-projection is defined as $\Pi_{\mathbf{c}}^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$, as demonstrated in Fig. 2 (b). Let us assume $\mathbf{p} \in \boldsymbol{\Omega}_i$ is a point in a reference image I_i with the inverse depth $d_{\mathbf{p}}$ that is observed at $\mathbf{p}'$ in a target image I_j. Suppose the poses of I_i and I_j are $\mathbf{T}_i$ and $\mathbf{T}_j$, respectively. Then the relationship between $\boldsymbol{p}$ and $\boldsymbol{p}'$ has the form

$$\boldsymbol{p}' = \Pi_{\boldsymbol{c}} \left(\mathbf{R}_{ij} \Pi_{\boldsymbol{c}}^{-1} \left(\boldsymbol{p}, d_{\boldsymbol{p}} \right) + \boldsymbol{t}_{ij} \right), \tag{1}$$

where $\mathbf{R}_{ij}$ and $\mathbf{t}_{ij}$ are the rotational and translational component of $\mathbf{T}_j \mathbf{T}_i^{-1}$.

Plücker Coordinates A 3D line $\boldsymbol{L}$ can be represented by the Plücker coordinates, as illustrated in Fig. 3 (a). The following formulas used in this paper are according to [28] and [35].

Let $\boldsymbol{\pi}$ denote the plane determined by $\boldsymbol{L}$ and the origin of the coordinate system, and $\tau_{\boldsymbol{L}}$ the distance from the origin to $\boldsymbol{L}$. Then the Plücker coordinates of $\boldsymbol{L}$ is a six-dimensional vector defined as

$$\boldsymbol{L} = [\boldsymbol{m}; \boldsymbol{d}], \quad s.t. \ ||\boldsymbol{d}||_2 = 1, \ \boldsymbol{d}^T \boldsymbol{m} = 0, \tag{2}$$

where $\boldsymbol{d}$ is the direction of $\boldsymbol{L}$, and $\boldsymbol{m}$ is referenced to as the moment vector, which is perpendicular to $\boldsymbol{\pi}$ with $||\boldsymbol{m}||_2 = \tau_{\boldsymbol{L}}$. The Plücker coordinates are homogeneous coordinates. Here we impose $||\boldsymbol{d}||_2 = 1$ on them to remove the scale ambiguity. Besides, this formulation can lead to simpler formulas in the following calculation.

Using the Plücker coordinates in (2), we can write the residual vector of the distance from a point $\boldsymbol{X} \in \mathbb{R}^3$ to $\boldsymbol{L}$ as

$$e\left(\boldsymbol{L}, \mathbf{X}\right) = \mathbf{m} - \mathbf{X} \times \mathbf{d}. \tag{3}$$

Given $\mathbf{R}$ and $\boldsymbol{t}$, the transformation matrix of the Plücker coordinates has the form

$$H\left(\mathbf{R}, \boldsymbol{t}\right) = \begin{bmatrix} \mathbf{R} & [\boldsymbol{t}]_{\times} \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}. \tag{4}$$

Using (4), we can write the transformation of the Plücker coordinates from one coordinate system to another as

$$\boldsymbol{L}' = H\left(\mathbf{R}, \boldsymbol{t}\right) \boldsymbol{L}. \tag{5}$$

The Plücker coordinates are convenient for computation. However, due to the constraints in (2), this representation is not convenient in optimization. In this paper, we represent the Plücker coordinates by two parameters, which leads to unconstrained optimization.

V. Direct Point-Line Model

A. Photometric Error

Direct method minimizes the photometric error. We adopt the photometric error model introduced in [1]. The photometric error for $\mathbf{p}$ with inverse depth $d_{\mathbf{p}}$ in I_i observed by I_j is defined as:

$$E_{\mathbf{p}_j} = \sum_{\mathbf{p} \in \mathbb{N}_{\mathbf{p}}} w_{\mathbf{p}} \left\| \left(I_j \left[\mathbf{p}' \right] - b_j \right) - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i \left[\mathbf{p} \right] - b_i \right) \right\|_{\gamma}, \tag{6}$$

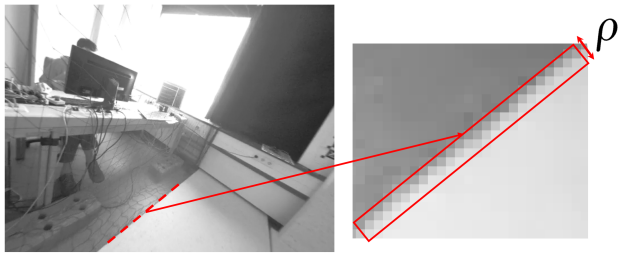


Fig. 4. Line support region. As introduced in [15], a real line segment has a support region due to the quantization or motion blur, which can represent the uncertainty of a line. We use this information to weigh the collinear constraints. ρ is the width of the line support region.

where a_i, a_j, b_i and b_j are the affine brightness transform parameters, t_i and t_j are the exposure time of I_i and I_j, $w_{\boldsymbol{p}}$ is a gradient-dependent weighting factor, $\mathbb{N}_{\boldsymbol{p}}$ is a set of neighborhoods around $\mathbf{p}$, and $\|\cdot\|_{\gamma}$ is the Huber norm. The relationship between $\mathbf{p}$ and $\mathbf{p}'$ is given in (1). Consequently, the full photometric error over all frames $\mathbb{F}$ has the form as

$$E_{photo} = \sum_{i \in \mathbb{F}} \sum_{\boldsymbol{p} \in \mathbb{P}_i} \sum_{j \in obs(\boldsymbol{p})} E_{\boldsymbol{p}_j}, \tag{7}$$

where $\mathbb{P}_i$ is the set of points in frame i, and $obs(\boldsymbol{p})$ is the set of frames where $\boldsymbol{p}$ is visible.

B. Collinear Constraint

We impose the collinear constraints on the points on lines. We first present a novel 3D line representation. Next, we introduce the cost function of the collinear constraint for the keyframes in the sliding window. Finally, we discuss the collinear prior yielded by the removed collinear points.

1) 3D Line Representation: We call the image where a 3D line $\boldsymbol{L}$ is first observed as the **anchor image** of $\boldsymbol{L}$. On the other hand, we call the subsequent images observing $\boldsymbol{L}$ as the **associate images**. Here we show that $\boldsymbol{L}$ in (2) can be parameterized by two unknowns in the coordinate system of the anchor image of $\boldsymbol{L}$, as demonstrated in Fig. 3 (b).

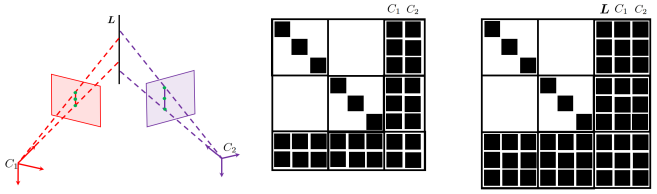
We assume the image has been undistorted and the camera matrix is $\mathbf{K}$, and suppose $\boldsymbol{l}_{\perp}$ is the captured image of $\boldsymbol{L}$ in the anchor image. The back-projection of $\boldsymbol{l}_{\perp}$ is a plane demonstrated in Fig. 2. Assume $\bar{\boldsymbol{e}}_1$ and $\bar{\boldsymbol{e}}_2$ are the homogeneous coordinates of the two endpoints of $\boldsymbol{l}_{\perp}$, and $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ are the corresponding 3D points on $\boldsymbol{L}$. Then we have $\boldsymbol{l}_{\perp} = \bar{\boldsymbol{e}}_1 \times \bar{\boldsymbol{e}}_2$, and the direction of $\boldsymbol{L}$ is $\boldsymbol{d} = \frac{\boldsymbol{E}_2 - \boldsymbol{E}_1}{\|\boldsymbol{E}_2 - \boldsymbol{E}_1\|_2}$. The norm of the back-projection plane $\boldsymbol{\pi}_{\boldsymbol{l}_{\perp}}$ of $\boldsymbol{l}_{\perp}$ in the camera coordinate system is $\boldsymbol{n} = \frac{\mathbf{K}^T \boldsymbol{l}_{\perp}}{||\mathbf{K}^T \boldsymbol{l}_{\perp}||_2}$. The direction of the back-projection ray of $\bar{\boldsymbol{e}}_1$ is $\boldsymbol{\alpha} = \frac{\mathbf{K}^{-1} \bar{\boldsymbol{e}}_1}{\left\| \mathbf{K}^{-1} \bar{\boldsymbol{e}}_1 \right\|_2}$. Assume $\theta_{\boldsymbol{L}}$ is the angle from $\boldsymbol{\alpha}$ to $\boldsymbol{d}$, and $\tau_{\boldsymbol{L}}$ is the distance from the camera origin to the 3D line. Then the Plücker coordinates $\boldsymbol{L}$ defined in (2) can be parameterized by $\theta_{\boldsymbol{L}}$ and $\tau_{\boldsymbol{L}}$ as below

$$\begin{aligned} \boldsymbol{m} &= \tau_{\boldsymbol{L}} \boldsymbol{n}, \\ \boldsymbol{d} &= cos\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\alpha} + sin\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\beta}, \end{aligned} \tag{8}$$

where $\boldsymbol{\beta} = \boldsymbol{n} \times \boldsymbol{\alpha}$.

2) Collinear Constraint from Anchor Image of $\boldsymbol{L}$: Suppose $\boldsymbol{p}_{\perp}^{\boldsymbol{L}}$ with depth $d_{\boldsymbol{p}_{\perp}^{\boldsymbol{L}}}$ is a point on $\boldsymbol{l}_{\perp}$ in the anchor image I_i of $\boldsymbol{L}$. Then the corresponding 3D point of $\boldsymbol{p}_{\perp}^{\boldsymbol{L}}$ in the coordinate system of I_i has the form $\boldsymbol{X}_{\perp}^{\boldsymbol{L}} = \Pi_{\mathbf{c}}^{-1} \left(\boldsymbol{p}_{\perp}^{\boldsymbol{L}}, d_{\boldsymbol{p}_{\perp}^{\boldsymbol{L}}} \right)$. Here $\boldsymbol{X}_{\perp}^{\boldsymbol{L}}$ and $\boldsymbol{L}$ are in the same coordinate system. Directly using (3), we get the collinear constraint on $\boldsymbol{X}_{\perp}^{\boldsymbol{L}}$ as below

$$E_{\boldsymbol{p}_{\perp}^{\boldsymbol{L}}}^{\boldsymbol{L}} = \left\| e\left(\boldsymbol{L}, \boldsymbol{X}_{\perp}^{\boldsymbol{L}} \right) \right\|_2^2. \tag{9}$$



(a) Photometric Error + Collinear Constraint (b) Hessian Matrix of Photometric Error (c) Hessian Matrix of Photometric Error + Collinear Constraint

Fig. 5. A schematic of the Hessian matrix of the photometric error (7) and the photometric error with the collinear constraint (15). 3 points are sampled from each 2D line. By properly aligning the variables, the Hessian matrix of (15) also has a sparse pattern. Thus the resulting linear system of the iterative algorithm can be efficiently solved [36].

Let us denote the set of sampled points on $\boldsymbol{l}_\perp$ as $\mathbb{A}^L$. The collinear constraint on the 3D points of $\mathbb{A}^{\boldsymbol{L}}$ is given by

$$E_\perp^{\boldsymbol{L}} = \sum_{\boldsymbol{p}_\perp^{\boldsymbol{L}} \in \mathbb{A}^L} E_{\boldsymbol{p}_\perp^{\boldsymbol{L}}}^{\boldsymbol{L}}. \tag{10}$$

3) Collinear Constraint from Associate Image of $\boldsymbol{L}$: Let us use $\boldsymbol{l}_j^{\boldsymbol{L}}$ to denote the 2D line of $\boldsymbol{L}$ in the associate image I_j. Assume $\boldsymbol{p}_j^{\boldsymbol{L}}$ is a pixel on $\boldsymbol{l}_j^{\boldsymbol{L}}$. Given its depth $d_{\boldsymbol{p}_j^{\boldsymbol{L}}}$, we can get the corresponding 3D point coordinates $\boldsymbol{X}_j^{\boldsymbol{L}} = \Pi_{\boldsymbol{c}}^{-1}\left(\boldsymbol{p}_j^{\boldsymbol{L}}, d_{\boldsymbol{p}_j^{\boldsymbol{L}}}\right)$ in the camera coordinate system of I_j. Here $\boldsymbol{X}_j^{\boldsymbol{L}}$ and $\boldsymbol{L}$ are in different coordinate systems. Using (3) and (5), we get the collinear constraint on $\boldsymbol{X}_j^{\boldsymbol{L}}$

$$E_{\boldsymbol{p}_j^{\boldsymbol{L}}}^{\boldsymbol{L}} = \left\| e\left(\boldsymbol{L}_j, \boldsymbol{X}_j^{\boldsymbol{L}}\right)\right\|_2^2, \ \boldsymbol{L}_j = H\left(\mathbf{R}_{ij}, \boldsymbol{t}_{ij}\right)\boldsymbol{L}, \tag{11}$$

where $\mathbf{R}_{ij}$ and $\boldsymbol{t}_{ij}$ are defined in (1). Assume $\mathbb{B}_j^L$ denotes the set of sampled points on $\boldsymbol{l}_j$. Then the constraint on the 3D points of $\mathbb{B}_j^{\boldsymbol{L}}$ has the form as

$$E_j^{\boldsymbol{L}} = \sum_{\boldsymbol{p}_j^{\boldsymbol{L}} \in \mathbb{B}_j^L} E_{\boldsymbol{p}_j^{\boldsymbol{L}}}^{\boldsymbol{L}}. \tag{12}$$

4) Collinear Constraints: As shown in Fig. 4, a line segment has a support region [15] due to the quantization or motion blur, which represents the uncertainty of the line segment. Let us denote the width of the support region as ρ. We use $\frac{1}{\rho}$ to weigh the collinear constraints. Unlike [18] that samples the same number of points for each line, we split the support region into several segments with a fixed length ϵ. In each segment, we select the point with the largest gradient. As a longer line generally has a lower uncertainty on its parameters, this sampling method can increase the weight of a longer line by introducing more points.

The 3D line $\boldsymbol{L}$ is observed in its anchor image and the subsequent associate images. Let $\mathbb{H}^L$ denote the set of associate images of $\boldsymbol{L}$. Using (10) and (12), we get the collinear constraint from $\boldsymbol{L}$ as

$$E^{\boldsymbol{L}} = \frac{1}{\rho_\perp^{\boldsymbol{L}}} E_\perp^{\boldsymbol{L}} + \sum_{j \in \mathbb{H}^L} \frac{1}{\rho_\perp^{\boldsymbol{L}} + \rho_j^{\boldsymbol{L}}} E_j^{\boldsymbol{L}}, \tag{13}$$

where $\rho_\perp^{\boldsymbol{L}}$ and $\rho_j^{\boldsymbol{L}}$ are the widths of the support region of the line segments in the anchor and associate images. Let $\mathbb{L}$ denote the set of all the 3D lines. Then the full cost function over $\mathbb{L}$ has the form

$$E_{line} = \sum_{\boldsymbol{L} \in \mathbb{L}} E^{\boldsymbol{L}}. \tag{14}$$

C. Full Cost Function

In our algorithm, the camera poses, 3D points and lines are jointly adjusted to minimize the photometric error as well as to meet the collinear constraint. Specifically, the full cost function over all frames, points and lines is the combination of (7) and (14)

$$E = E_{photo} + E_{line}, \tag{15}$$

Fig. 5 demonstrates the Hessian matrix of (15). It is of the general sparse pattern, thus the resulting linear system of the iterative algorithm can be efficiently solved [36].

D. Windowed Optimization

Minimizing the full cost (15) is computationally demanding. Thus we adopt the sliding window approach in [1] to balance the computational complexity and accuracy. For the removed keyframes, we adopt different strategies for E_{photo} and E_{line}.

1) Marginalization: For E_{photo}, we adopt the marginalization strategy used in [1]. Briefly, inactive keyframes and points are marginalized using the Schur complement, and all the remaining observations related to the inactive keyframes are removed to preserve the sparsity of the Hessian matrix. This leads to a quadratic function on the remaining unknowns that can be used in the subsequent optimization and marginalization. For 3D lines, we want to use the old observations to constrain the subsequent optimization, and update the line parameters when new observations emerge. Thus, we do not marginalize them. Instead, we introduce prior collinear constraints for a 3D line when its anchor image or associate images are removed from the sliding window.

2) Prior Collinear Constraint: We fix the poses of the keyframes removed from the sliding window and the depths of the collinear pixels in these keyframes. Given a fixed pose, we can transform the collinear points in this local camera coordinate system to the global coordinate system. These fixed points provide collinear priors to the corresponding lines in the subsequent windowed optimization. We denote the set of fixed points on $\boldsymbol{L}$ as $\mathbb{X}_f^{\boldsymbol{L}}$. Suppose $\boldsymbol{\chi}_k^{\boldsymbol{L}}$ is the kth point in $\mathbb{X}_f^{\boldsymbol{L}}$. Based on (3) and (5), the prior collinear constraint from $\boldsymbol{\chi}_k^{\boldsymbol{L}}$ on $\boldsymbol{L}$ has the form

$$E_{\boldsymbol{\chi}_k^{\boldsymbol{L}}}^{\boldsymbol{L}} = \left\| e\left(\boldsymbol{\chi}_k^{\boldsymbol{L}}, \boldsymbol{L}'\right)\right\|_2^2, \ \boldsymbol{L}' = H(\mathbf{R}_i^T, -\mathbf{R}_i^T \boldsymbol{t}_i)\mathbf{L}, \tag{16}$$

where $\mathbf{R}_i$ and $\boldsymbol{t}_i$ are the rotation matrix and translation vector from the global coordinate system to the coordinate system of the anchor image I_i of $\boldsymbol{L}$, respectively. Stacking the constraints in (16) for all $\boldsymbol{\chi}_k^{\boldsymbol{L}} \in \mathbb{X}_f^{\boldsymbol{L}}$, we get the collinear prior on $\boldsymbol{L}$

$$E_{prior}^{\boldsymbol{L}} = \left\| \boldsymbol{f}^L \right\|_2^2, \boldsymbol{f}^L = \left[\cdots, e\left(\boldsymbol{\chi}_k^{\boldsymbol{L}}, \mathbf{L}'\right)^T, \cdots\right]^T. \tag{17}$$

Suppose the size of $\mathbb{X}_f^L$ is $N^{\boldsymbol{L}}$. Then the size of $\boldsymbol{f}^{\boldsymbol{L}}$ is $3N^{\boldsymbol{L}}$.

3) Compressed Prior Collinear Constraint: As keyframes are removed, the size of $\boldsymbol{f}^{\boldsymbol{L}}$ in (17) enlarges, which seemingly inevitably increases the runtime of the optimization. Here we show that the computational complexity of the collinear prior in the optimization can be independent of $N^{\boldsymbol{L}}$. We first consider the form of $\boldsymbol{f}^{\boldsymbol{L}}$ and its Jacobian matrix in terms of the introduced line parameterization (8). We have the following lemma:

Lemma 1: For the parameterization (8) of $\boldsymbol{L}$, $\boldsymbol{f}^{\boldsymbol{L}}$ and its Jacobian matrix $\mathbf{J}_{\boldsymbol{f}^L}$ can be respectively factorized as follows

$$\boldsymbol{f}^{\boldsymbol{L}} = \mathbf{A}^{\boldsymbol{L}}\boldsymbol{L} \text{ and } \mathbf{J}_{\boldsymbol{f}^L} = \mathbf{A}^{\boldsymbol{L}}\mathbf{B}^{\boldsymbol{L}}, \tag{18}$$

where $\boldsymbol{L}$ is the Plücker coordinates given in (8), and $\mathbf{A}^{\boldsymbol{L}}$ and $\mathbf{B}^{\boldsymbol{L}}$ are defined in (25) and (23) with size $3N^{\boldsymbol{L}} \times 6$ and 6×2, respectively.

Let us define $\mathbf{M}^{\boldsymbol{L}} = (\mathbf{A}^{\boldsymbol{L}})^T\mathbf{A}^{\boldsymbol{L}}$. We have the following lemma for $\mathbf{M}^{\boldsymbol{L}}$:

Lemma 2: $\mathbf{M}^{\boldsymbol{L}}$ can be factorized as $\mathbf{M}^{\boldsymbol{L}} = \left(\mathbf{C}^{\boldsymbol{L}}\right)^T\mathbf{C}^{\boldsymbol{L}}$, where $\mathbf{C}^{\boldsymbol{L}}$ is a 6×6 matrix.

We define the compressed residual vector for $\boldsymbol{f}^{\boldsymbol{L}}$ as

$$\boldsymbol{g}^{\boldsymbol{L}} = \mathbf{C}^{\boldsymbol{L}}\boldsymbol{L}. \tag{19}$$

The size of $\boldsymbol{g}^{\boldsymbol{L}}$ is independent of the number of points in $\mathbb{X}_f^{\boldsymbol{L}}$. In the following theorem,we show that $\boldsymbol{g}^{\boldsymbol{L}}$ is sufficient for the Gauss-Netwon or LM algorithm.

Theorem 1: $\boldsymbol{g}^{\boldsymbol{L}}$ can replace $\boldsymbol{f}^{\boldsymbol{L}}$ in the Gauss-Newton or LM algorithm.

The computational complexity for calculating the compressed residual vector $\boldsymbol{g}^{\boldsymbol{L}}$, the Jaccobian matrix $\mathbf{J}_{\boldsymbol{g}^L}$, $\mathbf{J}_{\boldsymbol{g}^L}^T\mathbf{J}_{\boldsymbol{g}^L}$ and $\mathbf{J}_{\boldsymbol{g}^L}^T\boldsymbol{g}^{\boldsymbol{L}}$ are $\frac{6}{N^L}$ of the corresponding ones for $\boldsymbol{f}^L$. This can significantly reduce the computational cost when $N^{\boldsymbol{L}}$ is large as demonstrated in Fig. 1. We prove Lemma 1, 2 and Theorem 1 in the appendix.

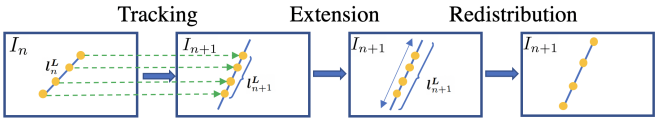


Fig. 6. Our line association between the keyframe I_n and I_{n+1} consists of three steps: tracking, extension and redistribution. In the tracking step, each point on line $\boldsymbol{l}_n^{\boldsymbol{L}}$ is individually tracked by minimizing the photometric error. Here we only require that the matched point in I_{n+1} is on the corresponding line, and we do not require an exact point-to-point correspondence. Then we seek to extend the line $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$ obtained from the tracking to reflect the new observation of the line in I_{n+1}. Finally, we resample some points from $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$.

4) Incremental Calculation of $\mathbf{M}^{\boldsymbol{L}}$: The most important step to get $\boldsymbol{g}^{\boldsymbol{L}}$ in (19) is to calculate $\mathbf{M}^{\boldsymbol{L}}$. The computational complexity will increase when $N^{\boldsymbol{L}}$ enlarges. Here we show that it is not necessary to compute $\mathbf{M}^{\boldsymbol{L}}$ from scratch every time. Instead, $\mathbf{M}^{\boldsymbol{L}}$ can be calculated incrementally. Assume I_k is removed from the sliding window and $N_k^{\boldsymbol{L}}$ 3D points on $\boldsymbol{L}$ are observed in I_k. We can calculate $\mathbf{A}_k$ for the new coming $N_k^{\boldsymbol{L}}$ 3D points by (22). Next we append $\mathbf{A}_k^{\boldsymbol{L}}$ to the current $\mathbf{A}^{\boldsymbol{L}}$ to generate the new one $\mathbf{A}_{new}^{\boldsymbol{L}} = [\mathbf{A}^{\boldsymbol{L}}; \mathbf{A}_k^{\boldsymbol{L}}]$. Then $\mathbf{M}_{new}^{\boldsymbol{L}}$ has the form

$$\begin{aligned}\mathbf{M}_{new}^{\boldsymbol{L}} &= (\mathbf{A}_{new}^{\boldsymbol{L}})^T \mathbf{A}_{new}^{\boldsymbol{L}} = (\mathbf{A}^{\boldsymbol{L}})^T \mathbf{A}^{\boldsymbol{L}} + \mathbf{A}_k^T \mathbf{A}_k \\ &= \mathbf{M}^{\boldsymbol{L}} + \mathbf{A}_k^T \mathbf{A}_k\end{aligned} \tag{20}$$

VI. Front End

The front end determines and initializes the points, lines and keyframe poses used in the back end. We introduce the line management into the front end of DSO, and slightly change the point and frame management of DSO.

A. Line Management

The line management includes line detection, initialization and association.

1) Line Detection: We adopt the LSD algorithm [15] to detect lines. Except for the first keyframe, the line detection is conducted after the line association described in the section VI-A3. Thus, starting from the second keyframe, we only detect lines in the regions which are not associated with the existing lines. As a line segment may be detected as several shorter ones, we seek to merge the line segments with similar parameters. Specifically, assume the two line segments $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ have the directions $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and the distances from the origin to $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ are c_1 and c_2. If $angle(\boldsymbol{v}_1, \boldsymbol{v}_2) < \vartheta$ and $|c_1 - c_2| < \varrho$, we calculate a new line $\boldsymbol{l}_3$ using all the points of $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. Here $angle(\cdot, \cdot)$ represents the angle between two vectors. If more than $q\%$ of the points are within a distance ζ to the new line, we merge $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. In our experiments, we set $\vartheta = 10°$, $\varrho = 10$ pixels, $q = 95$ and $\zeta = 2$ pixels. The new 3D lines are initialized as described below.

2) Line Initialization: Assume $\boldsymbol{l}_m^n$ is the mth new 2D line segment detected in the latest keyframe I_n. We sample N 2D points from it. In subsequent frames, the N points are then individually tracked along the epipolar line, and their depths are estimated as done for the ordinary point in DSO. After their depths are estimated, we check the quality of these depths. Specifically, given the depths, we calculate the 3D points using the back-projection function Π_c^{-1}. Then we conduct the principal component analysis (PCA) on these 3D points. Assume λ_1, λ_2 and λ_3 are the 3 resulting principal component variances in descending order. For an idea 3D line, we have $\lambda_2 = \lambda_3 = 0$. Thus, we consider the initialization succeeds if $\frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3} > \kappa$. If the initialization succeeds, we compute the parameters of the 3D line as introduced in section V-B1, and all the 3D points are projected onto the line and their depths are revised accordingly. We set $\kappa = 0.7$ in our experiments.

3) Line Association: We conduct line association for each initialized lines in a new keyframe I_{n+1}. Instead of adopting the feature-based line matching strategy [37], we introduce a **tracking-extension-redistribution** method to establish the line association among keyframes, as shown in Fig. 6.

Tracking: Let $\boldsymbol{l}_n^{\boldsymbol{L}}$ denote the image of the 3D line $\boldsymbol{L}$ in the nth keyframe I_n. Assume $\boldsymbol{p}_{n,k}^{\boldsymbol{L}}$ is the kth point on $\boldsymbol{l}_n^{\boldsymbol{L}}$. We compute the corresponding point of $\boldsymbol{p}_{n,k}^{\boldsymbol{L}}$ in I_{n+1} by (1), and denote it as $\hat{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}}$. If $\hat{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}}$ is out of I_{n+1}, it is ignored. Inspired by [22], we compute a correct $\delta\boldsymbol{u} \in \mathbb{R}^2$ for $\hat{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}}$ by minimizing the photometric error (6). Unlike [22], here the corrected point $\tilde{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}} = \hat{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}} + \delta\boldsymbol{u}$ does not have to match $\boldsymbol{p}_{n,k}^{\boldsymbol{L}}$. Instead, we only require $\tilde{\boldsymbol{p}}_{n+1,k}^{\boldsymbol{L}}$ can get closer to the line. Then we fit a line $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$ in I_{n+1} using these corrected points.

Extension: As a new part of $\boldsymbol{L}$ may be captured in I_{n+1}, we seek to extend $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$ to include the new observation. Motivated by [15], we introduce a directional region growing method to extend $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$. Assume the length of $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$ is $d_{n+1}^{\boldsymbol{L}}$. Let $\bar{g}$, δ and g_{min} denote the mean, standard deviation and minimum of the norms of the gradients at $\lceil d_{n+1}^{\boldsymbol{L}} \rceil$ points uniformly distributed on $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$, respectively. Here $\lceil \cdot \rceil$ represents the ceiling of a real number. Let $\boldsymbol{e}_{n+1}^{\boldsymbol{L}}$ denote one of the endpoints of $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$, and $\boldsymbol{\eta}_{n+1}^{\boldsymbol{L}}$ represent the direction of $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$. For each step, we first seek to extend $\boldsymbol{e}_{n+1}^{\boldsymbol{L}}$ with ν pixels. Specifically, we check the candidate endpoint $\boldsymbol{e}_{n+1}^{\boldsymbol{L}} = \boldsymbol{e}_{n+1}^{\boldsymbol{L}} + \nu\boldsymbol{\eta}_{n+1}^{\boldsymbol{L}}$. If the norm of the gradient at $\boldsymbol{e}_{n+1}^{\boldsymbol{L}}$ is larger than $max(\bar{g} - 2\delta, g_{min})$ and the angle between its gradient and the normal of $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$ is less than $22.5°$, $\boldsymbol{e}_{n+1}^{\boldsymbol{L}}$ is considered as the new endpoint of $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$. The threshold $22.5°$ is recommended by [15]. We continue this process until the test fails. We set $\nu = 5$ pixels in our experiments.

Redistribution: We sample N points from $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$. Let $\boldsymbol{p}_{n+1,k}^{\boldsymbol{L}}$ denote the kth point on $\boldsymbol{l}_{n+1}^{\boldsymbol{L}}$. The depth of $\boldsymbol{p}_{n+1,k}^{\boldsymbol{L}}$ is initialized as the depth of the 3D point on the back-projection ray of $\boldsymbol{p}_{n+1,k}^{\boldsymbol{L}}$ that is closest to $\boldsymbol{L}$.

B. Point and Frame Management

We slightly change the point and frame management in DSO [1]. For the point management, when selecting the candidate points, we avoid choosing the point whose distance to any of the lines is less than f pixels. For the frame management, we introduce a new criterion to determine whether a new keyframe is needed for lines. In detail, we add a new keyframe if there are more than k new lines have been initialized or the total length of the initialized lines exceeds γ pixels. We sample some points from each 2D line segment according to the length of the 2D line segment. For frame pose estimation, these collinear points are treated as ordinary points and combined with the ordinary points to get the pose of a new frame by direct image alignment. So lines do not impact on the runtime of the motion estimation. In our experiments, we set $f = 5$, $k = 3$ and $\gamma = 100$.

VII. Experimental Results

In this section, we compare the performance of our algorithm with the state-of-the-art VO or SLAM algorithms using the TUM monoVO dataset [38], ICL_NUM [39], EuRoC MAV Dataset [40] and TUM RGB-D dataset [41].

A. The TUM monoVO Dataset

The TUM monoVO dataset contains 50 photometrically calibrated and loop-closed sequences without the ground-truth pose. We run each sequence forwards and backwards 10 times to account for the nondeterministic behavior. As this dataset does not provide the ground-truth trajectory but has the loop-closure-ground-truth, we adopt the alignment error e_{align}, rotational drift e_r and scale drift e_s introduced in [38] to evaluate the performance. As in [1], we adopt the cumulative error plots to present the results, which reveal both accuracy and robustness of an algorithm by counting how many runs its errors are below a certain threshold.

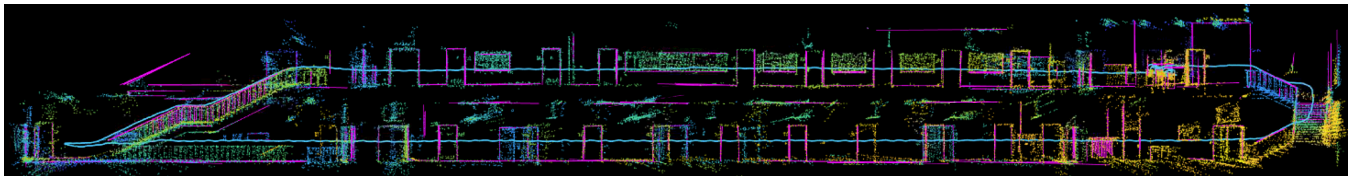


Fig. 7. The trajectory, lines and points generated by our algorithm on the TUM monoVO dataset [38].

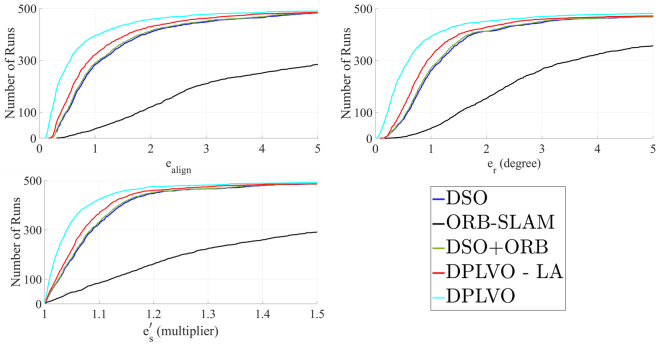


Fig. 8. The cumulative error of alignment error e_{align}, rotational drift e_r and the revised scale drift $e'_s = max(e_s, e_s^{-1})$ of different algorithms on the TUM monoVO dataset [38].

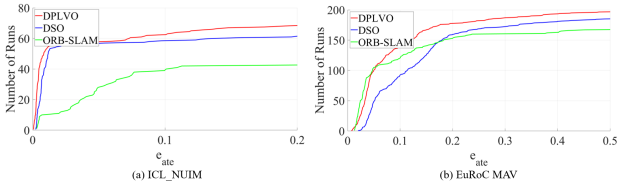


Fig. 9. The cumulative error of the absolute trajectory error e_{ate} of our algorithm, DSO [1] and ORB-SLAM [3] on the and ICL_NUM [39] dataset and the EuRoC MAV dataset [40].

Ablation Study We first use the TUM monoVO dataset [38] to evaluate the different design options of our algorithm. We compare the following two variants of our algorithm:

- **DPLVO - LA:** The line association is removed from our algorithm. That is to say this version does not include E_j^L defined in (12) in the cost function.
- **DSO + ORB:** We add the feature-based point matching for the corners and the resulting reprojection errors in DSO.

The results in Fig. 8 show that the collinear constraint alone can improve the performance of DSO. The proposed line association scheme can further increase the accuracy. The difference between the results of DSO and DSO + ORB is marginal. The optical flow tracking for the corner feature is well defined. Additional feature-based matching cannot significantly improve the performance. On the other hand, the number of edge points is generally much larger than the number of corner features in the man-made scenarios, as demonstrated in Fig. 1, and the photometric tracking for the points on the line is problematic. The collinear constraint and line association can alleviate this problem, thus they improve the results.

Comparison with the State-of-the-art We compare the performance of our algorithm with DSO [1] and ORB-SLAM [3]. Fig. 8 presents the results. It is clear that our algorithm outperforms DSO and ORB-SLAM. Our algorithm establishes long-term line association among keyframes which reduces the drift.

B. The EuRoC MAV and ICL_NUM Dataset

The ICL_NUM dataset [39] and EuRoC MAV dataset [40] consist of 11 and 8 sequences, respectively. In these sequences, the photometric calibration and the exposure time are not available. As the two datasets provide the ground-truth trajectories, we adopt the absolute trajectory error e_{ate} (ATE) to evaluate the performance. Again, we run each sequence 10 times forwards and backwards, and adopt the cumulative error plot to present the results in Fig. 9. The results show that our algorithm outperforms ORB-SLAM [3] and DSO [1] on the ICL_NUM dataset. ORB-SLAM achieves a better accuracy but less robust than DSO [1] and our algorithm. But our algorithm significantly reduces the gap between DSO and ORB-SLAM. One drawback of DSO is that it does not establish the accurate data correspondence. Our line association scheme overcomes this drawback, thus our algorithm improves the accuracy.

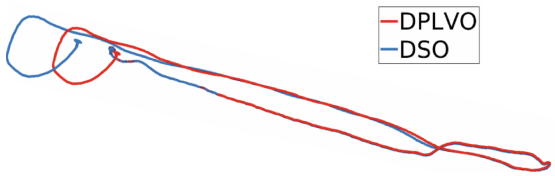


Fig. 10. The trajectories estimated by our algorithm and DSO [1] on sequence 40 of the TUM monoVO dataset [38]. The start point and the end point of this sequence are at the small place. The gap between the start and the end points of DSO is clearly larger than the one of our algorithm.

TABLE I
ABSOLUTE TRAJECTORY ERROR (CM) ON THE TUM RGBD BENCHMARK [41]. THE RESULTS ARE THE MEDIAN OVER 5 EXECUTIONS FOR EACH SEQUENCE.

Sequence	Ours	DLGO [18]	DSO [1]	PL-SLAM [5]	ORB-SLAM [3]
fr1_xyz	3.85	5.38	6.37	1.21	**0.9**
fr3_walk_xyz	9.7	27.5	15.5	1.54	**1.24**
fr3_walk_half	23.4	37.4	32.6	**1.60**	1.74

C. The TUM RGB-D Dataset

We also use the TUM RGB-D dataset [41] to evaluate the performance of our algorithm. The results are listed in table I. It is clear that our algorithm outperforms the state-of-the-art direct methods [1] and [18]. The TUM RGB-D dataset is challenging for the direct method. This dataset does not provide the photometric calibration and the exposure time. In addition, since the camera in the TUM RGB-D dataset is a rolling shutter camera, simply employing the direct method does not work well [42]. Thus the feature-based methods PL-SLAM [5] and ORB-SLAM surpass the direct methods in this dataset.

D. Runtime

We ran the runtime experiments on a laptop with an i7 3.4GHZ CPU and 16G memory. On the TUM monoVO dataset, the average runtime for the back end of our algorithm and DSO is 166.5 ms and 140.5 ms, respectively. Lines introduce more parameters, thus the time for optimization increases. The average runtime for the direct frame alignment, and point and frame management in the front end is 18.2 ms and 15.9 ms for our algorithm and DSO, respectively. Our algorithm treats the collinear points as ordinary points in the front

end, thus the runtime is similar. As our algorithm may introduce more points into these steps, thus the runtime is a bit longer. Our algorithm requires an additional 11.3 ms for the line management on average, but these computation only requires when a new keyframe is added. In addition, as the line management is independent of the point and frame management, we can run them in parallel. Therefore, the front end of our algorithm can achieve the real-time performance.

E. Qualitative Results

Fig. 7 demonstrates the 3D lines and point could generated by our algorithm. Fig. 10 shows the trajectories of our algorithm and DSO on sequence 40 of the TUM monoVO dataset. This sequence contains a loop. The camera returns back to the original location. The gap between the start and the finial points of DSO is larger than the one of our algorithm.

VIII. Conclusion

In this paper, we introduce the collinear constraint into DSO [1]. We establish long-term data association for lines. The collinear constraints in the removed keyframes are treated as priors in the following windowed optimization. We prove that no matter how many prior collinear constraints a line has, they can be compressed into six constraints. So the computation will not increase as new observations emerge. Furthermore, the resulting Hessian Matrix is still sparse, thus the optimization is efficient. Experimental results show that our algorithm significantly improves DSO and outperforms the state-of-the-art direct VO algorithms.

Appendix

A. Proof of Lemma 1

Proof. Let us first expand $e\left(\boldsymbol{\chi}_k^{\boldsymbol{L}}, \boldsymbol{L}'\right)$ in (16). Substituting the definition of H in (4) into (16), we have

$$\begin{aligned} e\left(\boldsymbol{\chi}_k^{\boldsymbol{L}}, \boldsymbol{L}'\right) &= \mathbf{R}_i^T \boldsymbol{m} - \left[\mathbf{R}_i^T \boldsymbol{t}_i\right]_\times \mathbf{R}_i^T \boldsymbol{d} - \left[\boldsymbol{\chi}_k^{\boldsymbol{L}}\right]_\times \mathbf{R}_i^T \boldsymbol{d} \\ &= \mathbf{R}_i^T \boldsymbol{m} - \underbrace{\left(\left[\mathbf{R}_i^T \boldsymbol{t}_i\right]_\times \mathbf{R}_i^T + \left[\boldsymbol{\chi}_k^{\boldsymbol{L}}\right]_\times \mathbf{R}_i^T\right)}_{\mathbf{U}_k} \boldsymbol{d} \\ &= \mathbf{R}_i^T \boldsymbol{m} - \mathbf{U}_k \boldsymbol{d} \end{aligned} \tag{21}$$

Substituting the line parameterization (8) into (21), we get

$$\begin{aligned} e_k\left(\tau_{\boldsymbol{L}}, \theta_{\boldsymbol{L}}\right) &= \tau_{\boldsymbol{L}} \mathbf{R}_i^T \boldsymbol{n} - cos\left(\theta_{\boldsymbol{L}}\right) \mathbf{U}_k \boldsymbol{\alpha} - sin\left(\theta_{\boldsymbol{L}}\right) \mathbf{U}_k \boldsymbol{\beta} \\ &= \underbrace{\left[\mathbf{R}_i^T, \quad -\mathbf{U}_k\right]}_{\mathbf{A}_k^{\boldsymbol{L}}} \underbrace{\begin{bmatrix} \tau_{\boldsymbol{L}} \boldsymbol{n} \\ cos\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\alpha} + sin\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\beta} \end{bmatrix}}_{\boldsymbol{L}} = \mathbf{A}_k^{\boldsymbol{L}} \boldsymbol{L}. \end{aligned} \tag{22}$$

Then we can calculate the Jacobian matrix $\mathbf{J}_k$ of $e_k\left(\tau_{\boldsymbol{L}}, \theta_{\boldsymbol{L}}\right)$. It has the form

$$\mathbf{J}_k^{\boldsymbol{L}} = \underbrace{\left[\mathbf{R}_i^T, \quad -\mathbf{U}_k\right]}_{\mathbf{A}_k^{\boldsymbol{L}}} \underbrace{\begin{bmatrix} \boldsymbol{n}, & \mathbf{0} \\ \mathbf{0}, & -sin\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\alpha} + cos\left(\theta_{\boldsymbol{L}}\right) \boldsymbol{\beta} \end{bmatrix}}_{\mathbf{B}^{\boldsymbol{L}}} = \mathbf{A}_k^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}} \tag{23}$$

Stacking all the residual vector from $\mathbb{X}_f^{\boldsymbol{L}}$, we can rewrite $\boldsymbol{f}^{\boldsymbol{L}}$ defined in (17) as

$$\boldsymbol{f}^{\boldsymbol{L}} = \underbrace{\begin{bmatrix} \cdots, & \left(\mathbf{A}_k^{\boldsymbol{L}}\right)^T, & \cdots \end{bmatrix}^T}_{\mathbf{A}^{\boldsymbol{L}}} \boldsymbol{L} = \mathbf{A}^{\boldsymbol{L}} \boldsymbol{L}. \tag{24}$$

Similarly, stacking the Jacobian matrix $\mathbf{J}_k^{\boldsymbol{L}}$ for $\boldsymbol{\chi}_k^{\boldsymbol{L}} \in \mathbb{X}_f^{\boldsymbol{L}}$, we get the Jacobian matrix of $\boldsymbol{f}^{\boldsymbol{L}}$

$$\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}} = \begin{bmatrix} \cdots, & \left(\mathbf{A}_k^{\boldsymbol{L}}\right)^T, & \cdots \end{bmatrix}^T \mathbf{B}^{\boldsymbol{L}} = \mathbf{A}^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}}. \tag{25}$$

□

B. Proof of Lemma 2

Proof. For any non-zero $\boldsymbol{x} \in \mathbb{R}^6$, we have $\boldsymbol{x}^T \mathbf{M}^{\boldsymbol{L}} \boldsymbol{x} = \boldsymbol{x}^T \left(\mathbf{A}^{\boldsymbol{L}}\right)^T \mathbf{A}^{\boldsymbol{L}} \boldsymbol{x} \geq 0$. Thus $\mathbf{M}^{\boldsymbol{L}}$ is a symmetric positive semi-definite matrix. Let us denote the eigendecomposition of $\mathbf{M}^{\boldsymbol{L}}$ as $\mathbf{Q}^T \boldsymbol{\Lambda} \mathbf{Q}$. Here $\boldsymbol{\Lambda}$ is a diagonal matrix. Denote the ith diagonal element of $\boldsymbol{\Lambda}$ as λ_i. As $\mathbf{M}^{\boldsymbol{L}}$ is positive semi-definite, we have $\lambda_i \geq 0$. Let us define $\mathbf{C} = \boldsymbol{\Lambda}^{\frac{1}{2}} \mathbf{Q}$, where $\boldsymbol{\Lambda}^{\frac{1}{2}}$ is a diagonal matrix whose ith diagonal element is $\sqrt{\lambda_i}$. It is easy to verify that $\mathbf{M}^{\boldsymbol{L}} = \left(\mathbf{C}^{\boldsymbol{L}}\right)^T \mathbf{C}^{\boldsymbol{L}}$. Here $\mathbf{C}$ is a 6×6 matrix, whose size is independent of the number of points. □

C. Proof of Theorem 1

Proof. Let us divide the residual vector $\boldsymbol{\delta}$ into two parts. One is $\boldsymbol{f}^{\boldsymbol{L}}$ defined in (17), and the other one is $\boldsymbol{h}$ containing the remaining residuals. We use $\boldsymbol{g}^{\boldsymbol{L}}$ to replace $\boldsymbol{f}^{\boldsymbol{L}}$ to form a compressed residual vector $\boldsymbol{\delta}^r$. Denote as $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}, \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}}$ and $\mathbf{J}_{\boldsymbol{h}}$ the Jacobian matrix of $\boldsymbol{f}^{\boldsymbol{L}}, \boldsymbol{g}^{\boldsymbol{L}}$ and $\boldsymbol{h}$, respectively. Then $\boldsymbol{\delta}$ and $\boldsymbol{\delta}^r$, and their corresponding Jacobian matrices $\mathbf{J}$ and $\mathbf{J}^r$ have the form:

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{f}^{\boldsymbol{L}} \end{bmatrix}, \ \boldsymbol{\delta}^r = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{g}^{\boldsymbol{L}} \end{bmatrix}, \ \mathbf{J} = \begin{bmatrix} \mathbf{J}_{\boldsymbol{h}} \\ \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}} \end{bmatrix}, \ \mathbf{J}^r = \begin{bmatrix} \mathbf{J}_{\boldsymbol{h}} \\ \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}} \end{bmatrix}. \tag{26}$$

The key step of the LM algorithm to minimize the least-squares problems $||\boldsymbol{\delta}||_2^2$ and $||\boldsymbol{\delta}^r||_2^2$ is to calculate the following linear equations

$$\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I} = \mathbf{J}^T \boldsymbol{\delta} \text{ and } \mathbf{J}^{rT} \mathbf{J}^r + \lambda \mathbf{I} = \mathbf{J}^{rT} \boldsymbol{\delta}^r. \tag{27}$$

For the Gauss-Newton algorithm, we set $\lambda = 0$. According to (26), $\mathbf{J}^T \mathbf{J}$ and $\mathbf{J}^{rT} \mathbf{J}^r$ can be written as

$$\mathbf{J}^T \mathbf{J} = \mathbf{J}_{\boldsymbol{h}}^T \mathbf{J}_{\boldsymbol{h}} + \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}, \ \mathbf{J}^{rT} \mathbf{J}^r = \mathbf{J}_{\boldsymbol{h}}^T \mathbf{J}_{\boldsymbol{h}} + \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}}. \tag{28}$$

Based on Lemma 1, the Jacobian matrices of $\boldsymbol{f}^{\boldsymbol{L}}$ and $\boldsymbol{g}^{\boldsymbol{L}}$ in the residual vectors $\boldsymbol{\delta}$ and $\boldsymbol{\delta}^r$ respectively have the form:

$$\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{A}^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}} & \cdots & \mathbf{0} \end{bmatrix}, \ \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{C}^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}} & \cdots & \mathbf{0} \end{bmatrix}. \tag{29}$$

Let us first focus on $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}$. Using (29), we have

$$\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \left(\mathbf{B}^{\boldsymbol{L}}\right)^T \left(\mathbf{A}^{\boldsymbol{L}}\right)^T \mathbf{A}^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}. \tag{30}$$

$\mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{g}}^{\boldsymbol{L}}$ has the same form as $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \mathbf{J}_{\boldsymbol{f}}^{\boldsymbol{L}}$. It only has one non-zero block $\left(\mathbf{B}^{\boldsymbol{L}}\right)^T \left(\mathbf{C}^{\boldsymbol{L}}\right)^T \mathbf{C}^{\boldsymbol{L}} \mathbf{B}^{\boldsymbol{L}}$. As $\left(\mathbf{C}^{\boldsymbol{L}}\right)^T \mathbf{C} = \left(\mathbf{A}^{\boldsymbol{L}}\right)^T \mathbf{A}$, we have $\left(\mathbf{J}_{\boldsymbol{f}}^{\boldsymbol{L}}\right)^T \mathbf{J}_{\boldsymbol{f}}^{\boldsymbol{L}} = \left(\mathbf{J}_{\boldsymbol{g}}^{\boldsymbol{L}}\right)^T \mathbf{J}_{\boldsymbol{g}}^{\boldsymbol{L}}$. So we have

$$\mathbf{J}^T \mathbf{J} = \mathbf{J}_r^T \mathbf{J}_r. \tag{31}$$

We next consider $\mathbf{J}^T \boldsymbol{\delta}$ and $\mathbf{J}^{rT} \boldsymbol{\delta}^r$. Substituting the definitions in (26) into $\mathbf{J}^T \boldsymbol{\delta}$ and $\mathbf{J}^{rT} \boldsymbol{\delta}^r$, we have

$$\mathbf{J}^T \boldsymbol{\delta} = \mathbf{J}_{\boldsymbol{h}}^T \boldsymbol{h} + \mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \boldsymbol{f}^{\boldsymbol{L}}, \ \mathbf{J}^{rT} \boldsymbol{\delta}^r = \mathbf{J}_{\boldsymbol{h}}^T \boldsymbol{h} + \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}}^T \boldsymbol{g}^{\boldsymbol{L}}. \tag{32}$$

Substituting (29) and (24) into $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \boldsymbol{f}^{\boldsymbol{L}}$, we get

$$\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \boldsymbol{f}^{\boldsymbol{L}} = \begin{bmatrix} \mathbf{0} & \cdots & (\mathbf{B}^{\boldsymbol{L}})^T (\mathbf{A}^{\boldsymbol{L}})^T \mathbf{A}^{\boldsymbol{L}} \boldsymbol{L} & \cdots & \mathbf{0} \end{bmatrix}. \tag{33}$$

Using (29) and the definition of $\boldsymbol{g}^{\boldsymbol{L}}$ in (19), we know that $\mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}} \boldsymbol{g}^{\boldsymbol{L}}$ has the same form as $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \boldsymbol{f}^{\boldsymbol{L}}$. Specifically, there only exists one non-zero block $(\mathbf{B}^{\boldsymbol{L}})^T (\mathbf{C}^{\boldsymbol{L}})^T \mathbf{C}^{\boldsymbol{L}} \boldsymbol{L}$. As $(\mathbf{C}^{\boldsymbol{L}})^T \mathbf{C}^{\boldsymbol{L}} = (\mathbf{A}^{\boldsymbol{L}})^T \mathbf{A}^{\boldsymbol{L}}$, we have $\mathbf{J}_{\boldsymbol{f}^{\boldsymbol{L}}}^T \boldsymbol{f}^{\boldsymbol{L}} = \mathbf{J}_{\boldsymbol{g}^{\boldsymbol{L}}} \boldsymbol{g}^{\boldsymbol{L}}$. Using (32), we get

$$\mathbf{J}^T \boldsymbol{\delta} = \mathbf{J}^{rT} \boldsymbol{\delta}^r. \tag{34}$$

Thus, using (27), (31) and (34), we know that $\boldsymbol{g}^{\boldsymbol{L}}$ can replace $\boldsymbol{f}^{\boldsymbol{L}}$ in the Gauss-Newton or LM algorithm. □

REFERENCES

[1] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[5] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 4503–4508.

[6] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust Visual SLAM with Point and Line Features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1775–1782.

[7] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3686–3693.

[8] S. J. Lee and S. S. Hwang, "Elaborate monocular point and line slam with robust initialization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1121–1129.

[9] R. Gomez-Ojeda, F.-A. Moreno, D. Zuñiga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, 2019.

[10] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Visual-Inertial Odometry with Point and Line Features," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2447–2454.

[11] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Lidar-monocular visual odometry using point and line features," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1091–1097.

[12] B. Fang and Z. Zhan, "A visual slam method based on point-line fusion in weak-matching scene," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, p. 1729881420904193, 2020.

[13] X. Li, Y. Li, E. P. Örnek, J. Lin, and F. Tombari, "Co-Planar Parametrization for Stereo-SLAM and Visual-Inertial Odometry," *IEEE Robotics and Automation Letters*, 2020.

[14] X. Li, Y. He, J. Lin, and X. Liu, "Leveraging planar regularities for point line visual-inertial odometry," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1775–1782.

[15] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2008.

[16] K. Zhao, Q. Han, C. Zhang, J. Xu, and M. Cheng, "Deep hough transform for semantic line detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 01, pp. 1–1, may 5555.

[17] S. Yang and S. Scherer, "Direct Monocular Odometry using Points and Lines," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3871–3877.

[18] S.-J. Li, B. Ren, Y. Liu, M.-M. Cheng, D. Frost, and V. A. Prisacariu, "Direct Line Guidance Odometry," in *2018 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2018, pp. 1–7.

[19] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct Sparse Odometry with Loop Closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2198–2204.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, pp. 1–17, 2021.

[21] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez, "PL-SVO: Semi-direct monocular visual odometry by combining points and line segments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4211–4216.

[22] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[23] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "Structslam: Visual slam with building structure lines," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1364–1375, 2015.

[24] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, "StructVIO: visual-inertial odometry with structural regularity of man-made environments," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 999–1013, 2019.

[25] Y. H. Lee, C. Nam, K. Y. Lee, Y. S. Li, S. Y. Yeon, and N. L. Doh, "VPass: Algorithmic compass using vanishing points in indoor environments," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 936–941.

[26] G. Zhang, D. H. Kang, and I. H. Suh, "Loop closure through vanishing points in a line-based monocular SLAM," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4565–4570.

[27] F. Camposeco and M. Pollefeys, "Using vanishing points to improve visual-inertial odometry," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 5219–5225.

[28] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] J. Sola, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular ekf-slam with points and lines," *International journal of computer vision*, vol. 97, no. 3, pp. 339–368, 2012.

[30] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[31] L. Zhang and R. Koch, "Structure and Motion from Line Correspondences: Representation, Projection, Initialization and Sparse Bundle Adjustment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 904–915, 2014.

[32] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer vision and image understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[33] D. G. Kottas and S. I. Roumeliotis, "Efficient and consistent vision-aided inertial navigation using line observations," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1540–1547.

[34] Y. Yang and G. Huang, "Aided inertial navigation: Unified feature representations and observability analysis," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3528–3534.

[35] A. Bartoli and P. Sturm, "The 3d line motion matrix and alignment of line reconstructions," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[36] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[37] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[38] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," in *arXiv:1607.02555*, July 2016.

[39] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2014, pp. 1524–1531.

[40] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

[42] D. Schubert, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "Direct sparse odometry with rolling shutter," in *European Conference on Computer Vision (ECCV)*, September 2018.