EDPLVO: Efficient Direct Point-Line Visual Odometry

Lipu Zhou[1], Guoquan Huang[1], Yinian Mao[1], Shengze Wang[2], and Michael Kaess[3]

***Abstract*— This paper introduces an efficient direct visual odometry (VO) algorithm using points and lines. Pixels on lines are generally adopted in direct methods. However, the original photometric error is only defined for points. It seems difficult to extend it to lines. In previous works, the collinear constraints for points on lines are either ignored [1] or introduce heavy computational load into the resulting optimization system [2]. This paper extends the photometric error for lines. We prove that the 3D points of the points on a 2D line are determined by the inverse depths of the endpoints of the 2D line, and derive a closed-form solution for this problem. This property can significantly reduce the number of variables to speed up the optimization, and can make the collinear constraint exactly satisfied. Furthermore, we introduce a two-step method to further accelerate the optimization, and prove the convergence of this method. The experimental results show that our algorithm outperforms the state-of-the-art direct VO algorithms.**

I. Introduction

Visual simultaneous localization and mapping (VSLAM) is a fundamental module for many robotic and computer vision applications, ranging from autonomous navigation to augmented reality (AR). The lightweight VSLAM system without loop closure is generally named as visual odomery (VO) [3], which is important for applications requiring real-time pose estimation on resource-limited embedded devices. Due to the importance of VSLAM and VO, they gain massive attention in computer vision and robotics community.

Nowadays, deep learning technology outperforms traditional methods in various computer visual tasks. In terms of VO, the learning-based methods have achieved significant progress in recent years [4]–[8]. However, as these methods require a powerful GPU, they are infeasible for real-time applications on embedded systems. Traditional VSLAM and VO systems are still more suitable for these applications. The traditional methods are generally classified into two categories, *i.e.,* feature-based (indirect) [9] and direct methods [1]. The feature-based method has dominated this field for a long time. Meanwhile, recent research [1], [2] shows that the direct method demonstrates high accuracy and robustness, even in low-textured scenarios that are generally challenging for the feature-based approaches. Thus this paper focuses on the direct method for VO.

[1]Lipu Zhou, Guoquan Huang and Yinian Mao are with Meituan, 7 Rongda Road, Chaoyang District, Beijing, 100012, China {zhoulipu,huangguoquan,maoyinian}@meituan.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

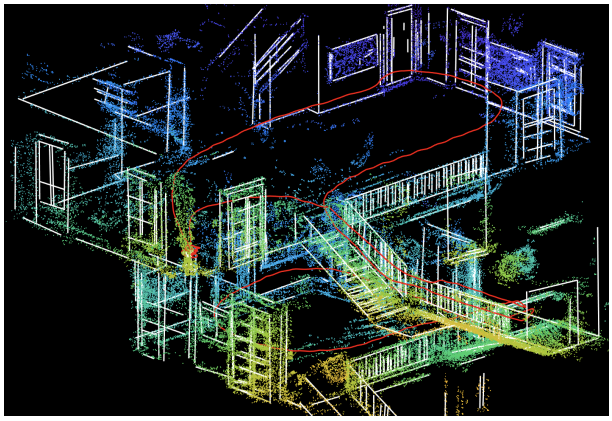


Fig. 1. The point cloud and lines generated by our algorithm.

Direct methods generally adopt pixels with sufficiently large gradients, typically including corners and points on lines. Points on lines can significantly outnumber corners in many man-made scenarios, as demonstrated in Fig. 2 (a). Tracking the corners through optical flow is well defined [10]. However, tracking points on lines is problematic, as there exists one-dimensional ambiguity along the line. Discarding the collinear constraint may lead to less accurate depth estimation, as illustrated in Fig. 2 (b). Although lines have been explored to overcome this problem, they generally significantly increase the computation load in the optimization [2], [11]. This work is based on our previous work, DPLVO [2], and we seek to accelerate the computation. The main contributions of this paper include:

- We extend the photometric error to lines. The original photometric error is only defined for points, which makes it difficult to be applied to lines. Instead of simply introducing the colllinear constraint into the cost function as done in [2], we propose a novel method to parameterize the 3D collinear points to make incorporating lines into the photometric error feasible. Specifically, we prove that the 3D points of any points on a 2D line are determined by the inverse depths of the two endpoints of the 2D line. This property can significantly reduce the number of variables. Meanwhile, our method has the collinear constraint exactly satisfied during the optimization, which can improve the accuracy.
- We introduce a two-step approach to limit the computational complexity due to introducing long-term line association into the optimization. In each iteration, we first fit 3D lines using the fixed inverse depths and keyframe poses. Then we use the new line parameters to regulate the optimization of the inverse depths and

keyframe poses. The two resulting optimization problems are easy to solve. We prove that this method can always converge.

Although this paper focuses on VO, the ideas can also be incorporated into the direct visual-inertial odometry (VIO) system [12].

II. Related Work

Points and lines widely exist in man-made scenarios. Feature-based and direct VSLAM or VO using points and lines have been extensively studied.

A. Line Matching

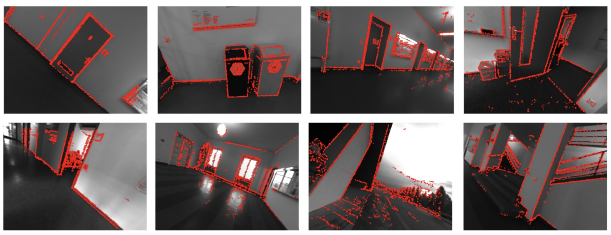
One of the challenges of using lines is to perform line matching. The descriptor-based line matching method is widely adopted in previous algorithms [13]–[16]. The line descriptor LBD [17] is generally used for this purpose. As the traditional line detection method, such as LSD [18], may be unstable, this may make the line matching fail [17]. Although deep learning based line detection methods [19]–[23] show promising results, these methods are generally computationally demanding. In addition, as the 3D line segment observed by a moving camera will change when a part of it emerges in or moves out the camera's field of view (FoV), this appearance change may also lead to the failure of matching. Recently, tracking-based solutions are proposed to overcome this problem. Wang *et al.* [17] propose the line flow that exploits the spatial and temporal coherence for line matching. In [2], the tracking-extension-redistribution method is presented for line matching. This method samples some points from a line, and then tracks them through minimizing photometric error along the epipolar line, which can be naturally incorporated into the front end of DSO [1]. Thus we adopt this method to establish the line association.

B. Feature-based Method

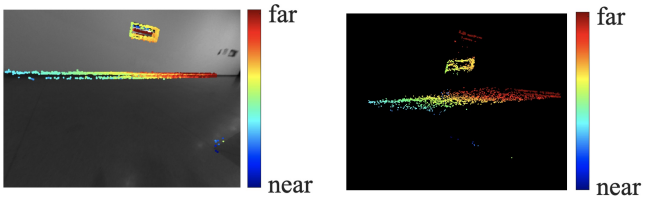
The performance of the feature-based methods that only use points [9], [24]–[26] decreases in the low-textured areas. Lines can complement points in the low-textured environments. The well established ORB-SLAM framework for points [9], [25], [26] can be easily extended to lines [13]. But this strategy significantly increases the computational load in both the front end and the back end [13]. The structure of the lines in the indoor environment can be used to improve the performance [27], [28]. Due to the Manhattan world assumption, these methods are difficult to be applied to outdoor scenarios.

C. Direct Method

Direct methods minimize the photometeric error to estimate camera poses and point depths [1], [29]. Drift is inevitable for a VO system. Some recent works [30], [31] introduce loop detection into the direct method for drift correction. Although the loop closure can correct the drift, it only rectifies the drift of the keyframe poses with hindsight. Accurate on-the-fly tracking poses are also important for many real-time applications, such as motion planning, control and AR. Lines provide another option to improve the performance.



(a) Man-made scenarios generally include massive collinear points. Collinearity can significantly reduce the number of variables.



(b) Less accurate depths of collinear points estimated by DSO. Collinearity can regulate the depth estimation.

Fig. 2. The necessity for employing the collinearity for the direct method. **(a)** exemplifies the points (marked in red) adopted in DSO [1] in different scenarios. Collinear points obviously outnumber corner features in these scenarios. Due to the ambiguity in matching, depths of collinear 2D points may be less accurate, as demonstrated in (**b**). Simply adding the collinear constraints into the cost as done in [2] will increase the computational complexity. This paper focuses on exploring the collinearity to improve the accuracy and reduce the computational complexity as well.

The photometeric error is to measure the difference between the intensity values of points in the reference and target images. **Unlike the feature-based methods where the geometrical distance can be defined for difference types of features, the photometric error is only defined for points. Thus the direct method is hard to be extended for lines.** In the literature, the collinear constraint is generally used to regulate the depth estimation. In [11] and [32], a 3D line is fitted to collinear points, and then the collinear points are projected into this line. Lines are not jointly optimized with points and keyframe poses in these methods. In [2], the collinear constraint is combined with the photometeric error to jointly optimize 3D lines with points and keyframe poses. But there is no guarantee that the collinear constraint can be exactly satisfied during the optimization, and the lines significantly increase the computational load of the optimization. In addition, the 3D line in [2] is represented in the back-projected plane of its first 2D line observation with two degrees of freedom (DoF). Although this parameterization reduces the number of unknowns to speed up the optimization, the estimation error of the 2D line may result in a suboptimal result. Furthermore, the 3D lines in [2] are initialized by fitting to the collinear 3D points, which are initially estimated without considering the collinear constraint. If the collinear points are of bad quality, as demonstrated in Fig. 2 (b), the 3D line initialization will be inaccurate or even fail, so that the 3D line may degrade or lose its ability to regulate the depth estimation of the collinear points. In this paper, we seek to overcome these drawbacks and show how to efficiently optimize the full four-DoF of the 3D lines with points and keyframe poses.

III. 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.

Photometric Error The photometric error is defined between a reference image I_i and a target image I_j. We represent a camera pose by a rotation matrix $\mathbf{R} \in SO_3$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$, or more concisely by a transformation matrix $\mathbf{T} \in SE(3)$. Let us denote the poses of the reference and the target images as $\mathbf{T}_i$ and $\mathbf{T}_j$, respectively. Let Ω represent the image domain. Suppose $\boldsymbol{x} \in \Omega$ is a point in I_i with the inverse depth d. Assume $\boldsymbol{x}$ is also observed by I_j at $\boldsymbol{x'} \in \Omega_j$. The relationship between $\boldsymbol{x}$ and $\boldsymbol{x'}$ has the form

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

where $\Pi_{\boldsymbol{c}} : \mathbb{R}^3 \to \Omega$ and $\Pi_{\boldsymbol{c}}^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$ denote the projection and back-projection function with the camera intrinsic parameters $\boldsymbol{c}$, and $\mathbf{R}_{ij}$ and $\boldsymbol{t}_{ij}$ are the rotational and translational components of $\mathbf{T}_j \mathbf{T}_i^{-1}$, respectively. We adopt the photometric error formulated in [1] with the form as:

$$E_{\boldsymbol{x}j} = \sum_{\boldsymbol{x} \in \mathbb{N}_{\boldsymbol{x}}} w_{\boldsymbol{x}} \left\| \left(I_j \left[\boldsymbol{x}' \right] - b_j \right) - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i \left[\boldsymbol{x} \right] - b_i \right) \right\|_{\gamma}, \tag{2}$$

where t_i and t_j are the exposure time of I_i and I_j, a_i, a_j, b_i and b_j are the affine brightness transform parameters, $w_{\boldsymbol{x}}$ is a gradient-dependent weighting factor, $\mathbb{N}_{\boldsymbol{x}}$ is a set of neighborhoods around $\boldsymbol{x}$, and $\|\cdot\|_{\gamma}$ is the Huber norm. The relationship between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is given in (1).

Plücker Coordinates A 3D line can be represented by the Plücker coordinates [33]. Given two points $\mathbf{p}_1$ and $\mathbf{p}_2$ on a 3D line, the Plücker coordinates of the 3D line have the form:

$$\boldsymbol{L} = [\boldsymbol{m}; \boldsymbol{d}], \ \boldsymbol{m} = \mathbf{p}_1 \times \boldsymbol{d} \text{ and } \boldsymbol{d} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|_2}, \tag{3}$$

where $\times$ represents the cross product. The Plücker coordinates are homogeneous coordinates. Here we normalize $\boldsymbol{d}$, as this can lead to more concise formulas in the following description. Using the Plücker coordinates in (3), we can write the distance from a point $\mathbf{p}$ to $\boldsymbol{L}$ as

$$e\left(\boldsymbol{L}, \mathbf{p}\right) = \boldsymbol{m} - \mathbf{p} \times \boldsymbol{d}. \tag{4}$$

Due to the overparamerization, the Plücker coordinates are seldom directly used in optimization. Instead, a certain parameterization for the Plücker coordinates is generally adopted in the optimization [34]–[36]. But they generally suffer from singularity under certain configurations. Our algorithm described in section IV-F can avoid parameterizing the 3D line.

IV. Direct Point-Line Model

A. Parameterize 3D Points of Collinear 2D Points

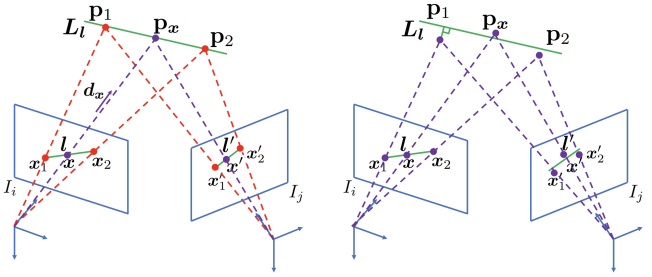
In this section, we show that the 3D point of a 2D point on a 2D line can be determined by the inverse depths of the two endpoints of the 2D line, as demonstrated in Fig. 3 (a). Formally, assume $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the endpoints of a 2D line segment $\boldsymbol{l}$ with the inverse depths d_1 and d_2, respectively. Using the back-projection function $\Pi_{\boldsymbol{c}}^{-1}$ introduced in (1), the 3D points for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have the form

$$\mathbf{p}_1 = \Pi_{\boldsymbol{c}}^{-1}(\boldsymbol{x}_1, d_1) \text{ and } \mathbf{p}_2 = \Pi_{\boldsymbol{c}}^{-1}(\boldsymbol{x}_2, d_2). \tag{5}$$

Using (3), we can compute the Plücker coordinates $\boldsymbol{L_l}$ of the 3D line determined by $\mathbf{p}_1$ and $\mathbf{p}_2$. Let us write $\boldsymbol{L_l}$ as

$$\boldsymbol{L_l} = [\boldsymbol{m_l}; \boldsymbol{d_l}]. \tag{6}$$

Note that here $\boldsymbol{L_l}$ is a function of d_1 and d_2.



(a) Photometric error for a line (b) Collinear constraint in DPLVO [2]

Fig. 3. A schematic of the photometric error for a line introduced in this paper (**a**) and the collinear constraint used in DPLVO [2] (**b**). In (**a**), the 3D point $\boldsymbol{p_x}$ of a 2D point $\boldsymbol{x}$ on the 2D line $\boldsymbol{l}$ are determined by the inverse depths of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The $\boldsymbol{p}_1$, $\boldsymbol{p_x}$ and $\boldsymbol{p}_2$ and their projections $\boldsymbol{x}'_1$, $\boldsymbol{x}'$, and $\boldsymbol{x}'_2$ on I_j are exactly collinear. In fact, no matter how many points are sampled from $\boldsymbol{l}$, the number of variables will not increase and the collinearity among them will be exactly satisfied. In (**b**), a 3D line $\boldsymbol{L_l}$ is explicitly introduced to impose collinear constraints on $\boldsymbol{p}_1$, $\boldsymbol{p_x}$ and $\boldsymbol{p_2}$. There is no guarantee that these points and their projections on I_j are collinear. Furthermore, enlarging the number of sampled points on $\boldsymbol{l}$ will increase the number of variables.

Suppose $\boldsymbol{x}$ is a point on $\boldsymbol{l}$, which is different from $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. According to [33], given the intrinsic camera matrix $\mathbf{K}$, the direction of the back-projected ray $\boldsymbol{L_x}$ of $\boldsymbol{x}$ has the form

$$\boldsymbol{d_x} = \frac{\mathbf{K}^{-1}\bar{\boldsymbol{x}}}{\|\mathbf{K}^{-1}\bar{\boldsymbol{x}}\|_2}, \tag{7}$$

where $\bar{\boldsymbol{x}}$ is the homogeneous coordinates of $\boldsymbol{x}$. Let us denote the 3D point of $\boldsymbol{x}$ as $\mathbf{p_x}$. Given d_1 and d_2, the following theorem gives a closed-form solution for $\mathbf{p_x}$.

Theorem 1: The 3D point $\mathbf{p_x}$ of a 2D point $\boldsymbol{x}$ on a 2D line $\boldsymbol{l}$ is determined by d_1 and d_2 with the form:

$$\begin{aligned} \mathbf{p_x} &= \left(\mathbf{A_l}^T \mathbf{A_l} \right)^{-1} \mathbf{A_l}^T \boldsymbol{b_l}, \\ \mathbf{A_l} &= \begin{bmatrix} [\boldsymbol{d_l}]_{\times} \\ [\boldsymbol{d_x}]_{\times} \end{bmatrix} \text{ and } \boldsymbol{b_l} = - \begin{bmatrix} \boldsymbol{m_l} \\ \mathbf{0}_{3\times 1} \end{bmatrix}, \end{aligned} \tag{8}$$

where $[\boldsymbol{d_l}]_{\times}$ and $[\boldsymbol{d_x}]_{\times}$ are the skew-symmetric matrices of $\boldsymbol{d_l}$ and $\boldsymbol{d_x}$ that are defined in (6) and (7), respectively.

We prove Theorem 1 in the Appendix.

B. Photometric Error for Lines

The photometric error introduced in (2) is just defined for points. In this section, we show that lines can be easily incorporated into the photometric error (2), as demonstrated in Fig. 3 (a).

Assume that a 3D line $\boldsymbol{L}$ is observed by a reference image I_i and a target image I_j, and the image of $\boldsymbol{L}$ in I_i is $\boldsymbol{l}$. We sample some points from $\boldsymbol{l}$ as done in [2]. According to Theorem 1, it is clear that, given d_1 and d_2, the 3D points of the internal points on $\boldsymbol{l}$ are determined. Thus we use different formulas to calculate the photometric errors for the endpoints and the internal points of $\boldsymbol{l}$.

For an internal point $\boldsymbol{x}$, we first use (8) to calculate the corresponding 3D point $\mathbf{p_x}$. Then we project $\mathbf{p_x}$ into I_j. This process can be formulated as

$$\boldsymbol{x}' = \Pi_{\mathbf{c}}(\mathbf{R}_{ij}\mathbf{p_x} + \mathbf{t}_{ij}), \tag{9}$$

where $\mathbf{R}_{ij}$ and $\mathbf{t}_{ij}$ are the rotation matrix and the translation vector from the coordinate system of I_i to the coordinate system of I_j and are defined in (1). We can substitute (9) into (2) to get the photometric error $E^{\boldsymbol{l}}_{\boldsymbol{x}j}$ for the internal point $\boldsymbol{x}$ of $\boldsymbol{l}$.

For the endpoints $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ of $\boldsymbol{l}$, we adopt (2) to directly calculate their photometric errors $E_{\boldsymbol{x}_1 j}$ and $E_{\boldsymbol{x}_2 j}$, respectively.

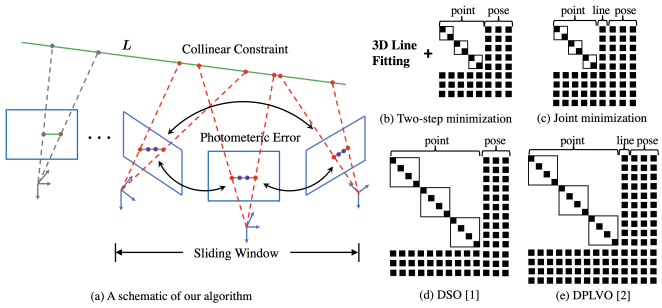


Fig. 4. A schematic of our algorithm for one 3D line and the sliding window of size three (**a**) and the Hessian matrices of different methods (**b**)-(**e**). Four points are sampled from each 2D line. As done in DPLVO [2], the invisible points of an active 3D line are fixed as priors in the following optimization (demonstrated as the gray points in (**a**)). (**b**) and (**c**) demonstrate the Hessian matrices of our two-step minimization algorithm and directly minimizing (12), respectively. For our algorithm, no matter how many points are sampled from a 2D line, only the inverse depths of the two endpoints are optimized, since the 3D points of other points on the 2D line are determined by them, as demonstrated in Fig. 3 (a). (**d**) and (**e**) demonstrate the Hessian matrices of DSO [1] and DPLVO [2], respectively. DSO only considers the photometric error of each point, and DPLVO imposes collinear constraints on collinear points. Both methods introduce an inverse depth for each pixel. It is clear that our two-step minimization algorithm leads to the smallest Hessian matrix.

Assume that N internal points of $\boldsymbol{l}$ are sampled, which forms a set $\mathbb{X}$. We formulate the photometric error for a 2D line $\boldsymbol{l}$ as

$$E_{\boldsymbol{l}j} = E_{\boldsymbol{x}_1 j} + E_{\boldsymbol{x}_2 j} + \sum_{\boldsymbol{x}\in\mathbb{X}} E^{\boldsymbol{l}}_{\boldsymbol{x}j}. \tag{10}$$

Note that $E_{\boldsymbol{l}j}$ only depends on the inverse depths of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, no matter how many points are sampled from $\boldsymbol{l}$.

C. Collinear Constraint for Line Association

In the above section, we incorporate lines into the photometric error. As an unbounded object, a 3D line can exist in the FoV of a camera for a long time. We adopt the method introduced in [2] to establish the line association, and introduce the collinear constraint to regulate the depths of the endpoints, as demonstrated in Fig. 4.

Suppose that a 3D line with the Plücker coordinates $\boldsymbol{L}$ is observed by $M_{\boldsymbol{L}}$ poses whose indices form a set $\mathbb{O}_{\boldsymbol{L}}$. Let us denote the pose with index $i \in \mathbb{O}_{\boldsymbol{L}}$ as $\mathbf{T}_i$, and the 2D line observation of $\boldsymbol{L}$ at $\mathbf{T}_i$ as $\boldsymbol{l}_i$. We use $\mathbf{p}_{i,1}$ and $\mathbf{p}_{i,2}$ to represent the two 3D endpoints for $\boldsymbol{l}_i$. $\mathbf{p}_{i,1}$ and $\mathbf{p}_{i,2}$ can be computed by (5) and are in the local camera coordinate system. Using $\mathbf{T}_i^{-1}$, we can transform $\mathbf{p}_{i,1}$ and $\mathbf{p}_{i,2}$ into the global coordinate system. Let us denote the global coordinates of $\mathbf{p}_{i,1}$ and $\mathbf{p}_{i,2}$ as $\mathbf{q}_{i,1}$ and $\mathbf{q}_{i,2}$, respectively. This work adopts the LSD algorithm [18] for line detection. Due to noise, quantization error and motion blur, each line detected by the LSD algorithm is associated with a line support region. Let ρ_i represent the width of the support region of $\boldsymbol{l}_i$. ρ_i can reflect the uncertainty of the endpoints of $\boldsymbol{l}_i$, and in turn impact the uncertainty of the 3D endpoints $\mathbf{q}_{i,1}$ and $\mathbf{q}_{i,2}$. A larger ρ_i generally means a higher uncertainty on the 3D endpoints. Thus, we adopt $\frac{1}{\rho_i}$ to weigh the collinear constraint. Using (4) and the above notations, we formulate the collinear constraints for $\boldsymbol{L}$ as:

$$E_{\boldsymbol{L}} = \sum_{i\in\mathbb{O}_{\boldsymbol{L}}} \frac{1}{\rho_i} \left(\|e\left(\boldsymbol{L}, \boldsymbol{q}_{i,1}\right)\|_2^2 + \|e\left(\boldsymbol{L}, \boldsymbol{q}_{i,2}\right)\|_2^2 \right), \tag{11}$$

where $e\left(\boldsymbol{L}, \boldsymbol{q}_{i,j}\right)$ $(j = 1, 2)$ is defined in (4), representing the distance between $\boldsymbol{L}$ and $\boldsymbol{q}_{i,j}$.

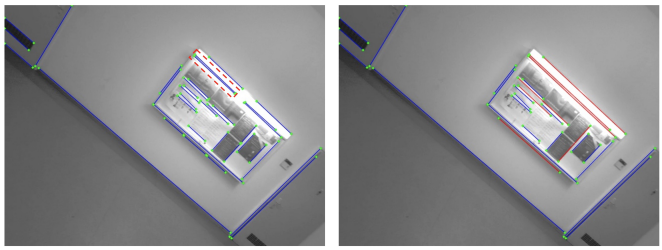
D. Model Formulation

Our model combines the photometric errors from points and lines, and the collinear constraints from line association. Using (2), (10) and (11), the full cost over all points, lines and keyframes is formulated as

$$E = \underbrace{\sum_{i\in\mathbb{F}} \sum_{\boldsymbol{p}\in\mathbb{P}_i} \sum_{j\in obs(\boldsymbol{p})} E_{\boldsymbol{p}j}}_{point\ photo} + \underbrace{\sum_{i\in\mathbb{F}} \sum_{\boldsymbol{l}\in\mathbb{L}_i} \sum_{j\in obs(\boldsymbol{l})} E_{\boldsymbol{l}j}}_{line\ photo} + \underbrace{\sum_{\boldsymbol{L}\in\mathbb{L}} E_{\boldsymbol{L}}}_{collinearity}, \tag{12}$$

where i runs over all keyframes $\mathbb{F}$, $\mathbb{P}_i$ and $\mathbb{L}_i$ are the sets of 2D points and 2D lines in keyframe i, $obs(\boldsymbol{p})$ and $obs(\boldsymbol{l})$ are the sets of keyframes where $\boldsymbol{p}$ and $\boldsymbol{l}$ are visible, and $\mathbb{L}$ is the set of 3D lines. For $obs(\boldsymbol{l})$, we consider a line segment $\boldsymbol{l}$ is visible at a keyframe j, if any sampled points on $\boldsymbol{l}$ are visible at the keyframe j.

Note that the collinear constraint plays a different role in DPLVO [2] and this paper. The collinear constraint in DPLVO [2] is used to regulate the depths of all the 2D points sampled from the matched 2D lines. There is no guarantee that 3D points from a 2D line will be collinear. In this paper, each 2D line segment in a keyframe has a corresponding 3D line segment estimated by minimizing the line photometric error (10). That is to say even without the collinearity term in (12), the collinearity of the 3D points from each 2D line can still be satisfied. Thus, in this work, the collinear constraints are only imposed on the endpoints of the 3D line segments to make them consistent.



(a) Lines detected by LSD (b) Lines after combination

Fig. 5. Results of line combination. (a) illustrates lines detected by the LSD algorithm [18]. It is clear that some line segments belong to a same line. Line segments with similar parameters are merged, as demonstrated by the red lines in (b). Note that we do not combine adjacent parallel lines, as marked in the red box in (a). In this case, the merged line will not lie on the edge. Instead, it may lie in a low-gradient area between the two lines.

E. Windowed Optimization

We employ the sliding window strategy to optimize the full cost (12) to balance accuracy and efficiency, as demonstrated in Fig. 4. Marginalization is generally adopted in previous direct VO algorithms [1], [2]. As a 3D line can exist in the FoV of a camera for a long time, we want to keep updating a line when it is visible. In addition, as wrong line associations may be included in (12), we want to remove the wrong line association when we find the point-to-line distance (4) is too large during the optimization. The wrong association is difficult to be removed once it is marginalized out. Thus this work does not adopt the marginalization method. Instead, we adopt the optimization strategy that is similar to the local bundle adjustment described in [9]. Specifically, we refine active poses within the sliding window and the visible points and lines. Other parameters in (12) are fixed. Let us take the term $E_{\boldsymbol{L}}$ defined in (11) as an example. If $\boldsymbol{q}_{i,1}$ and $\boldsymbol{q}_{i,2}$ become invisible, they get fixed, as demonstrated by the gray points in Fig. 4 (a). Thus, $E_{\boldsymbol{L}}$ becomes a function that only depends on $\boldsymbol{L}$. For a long 3D line segment, this strategy may introduce many prior collinear constraints [2]. We employ the method introduced in [2] to speed up the computation. Wrong associations and large photometric errors can be easily removed in this framework. We adopt the idea introduced in [37] to efficiently solve the resulting optimization problem, where the Schur complement is constructed incrementally and a residual is relinearized if the change of the involved parameters is large.

Algorithm 1: Two-step minimization for E in (12).

while *not converge* **do**

1) Use the latest poses and inverse depths to fit 3D lines $\hat{\mathbb{L}} = \{\hat{\boldsymbol{L}} | \hat{\boldsymbol{L}} = \arg\min_{\boldsymbol{L}} E_{\boldsymbol{L}}, \boldsymbol{L} \in \mathbb{L}\}$;
2) Fix $\hat{\mathbb{L}}$ and conduct one Levenberg-Marquardt step [38] to update poses and inverse depths to reduce the cost E;

end

F. Two-step Minimization

Lines enlarge the number of unknowns and correlate with points and poses. Thus jointly optimizing lines with points and poses will increase the computational load. We introduce a two-step approach to minimize (12). Let us first focus on the last term of (12). Points, lines and poses are correlated in this term. Given the camera poses and the inverse depths of the 2D endpoints, this problem is equivalent to fitting 3D lines to sets of points, *i.e.*, $\hat{\mathbb{L}} = \{\hat{\boldsymbol{L}} | \hat{\boldsymbol{L}} = \arg\min_{\boldsymbol{L}} E_{\boldsymbol{L}}, \boldsymbol{L} \in \mathbb{L}\}$, where $E_{\boldsymbol{L}}$ is defined in (11). On the other hand, if the line parameters in $\hat{\mathbb{L}}$ are fixed, the resulting cost function E for poses and inverse depths can be minimized as efficiently as only considering the photometric error alone. We can iterate these two steps. The algorithm, named two-step minimization, is summarized in Algorithm 1. Note that one Levenberg-Marquardt (LM) step may include more than one iteration to reduce the cost [38] . One question is whether the two-step minimization algorithm will converge or not. The following theorem gives an answer to this question.

Theorem 2: The two-step minimization algorithm always converges.

We prove Theorem 2 in the Appendix.

V. Front-end

Our front-end is similar to DPLVO [2], which manages points, lines and frames. The main differences lie in 3D line initialization and 2D line combination.

3D Line Initialization In DPLVO [2], some points are sampled from a new 2D line. Then their 3D points are estimated individually without considering the collinear constraint. The 3D line is initialized by fitting to these 3D points. If the 3D points are less accurate as demonstrated in Fig. 2 (b), the line initialization will be degraded or even fail, which impacts the stability of the algorithm. In this paper, we consider imposing the collinearity at the line initialization step. Specifically, given a new 2D line $\boldsymbol{l}$, we track the sampled 2D points in a subsequent image along the epipolar line by minimizing the photometric error, as done in DPLVO. This forms a set of correspondences $\{\boldsymbol{x} \leftrightarrow \boldsymbol{x}'\}$. This tracking is unlikely to generate accurate point-point correspondences, due to the pose error and the ambiguity along the line $\boldsymbol{l}$. But it can generate accurate line-line correspondences. We fit a 2D line $\boldsymbol{l}'$ to the tracked points $\{\boldsymbol{x}'\}$. Assume $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ are the back-projected planes of $\boldsymbol{l}$ and $\boldsymbol{l}'$, respectively. We calculate the 3D line by line triangulation [17] if the angle between $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ is larger than 3°. Then the inverse depths of the endpoints of $\boldsymbol{l}$ can be computed and then used in the subsequent line tracking. Motivated by [17], when a new line association is available, we update the 3D endpoints by averaging. Lines with large photometric errors during tracking will be removed, and the points sampled from these lines will then be treated as normal points in DSO [1].

2D Line Combination The LSD line detector [18] may return several segments of a 2D line, as demonstrated in Fig. 5 (a). In DPLVO [2], nearby lines with similar parameters are merged. This approach may integrate adjacent but different lines, as marked in the red dashed rectangle in Fig. 5 (a). In this case, the merged line may lie in an area with low gradient. Thus we do not combine two lines, when the projections to the merged line are overlapping or the angle between the mean gradients of the two lines is large. As demonstrated in Fig. 5, the merged lines may pass through low-gradient areas. Thus, the original segments are projected to the merged line. The 2D points are sampled from the projected line segments separately. If approximately parallel lines are too close to each other, we sum up the magnitude of the gradient along the line, and keep the one with the largest amassed magnitude. We also try to merge newly detected lines with the existing ones to avoid introducing new parameters.

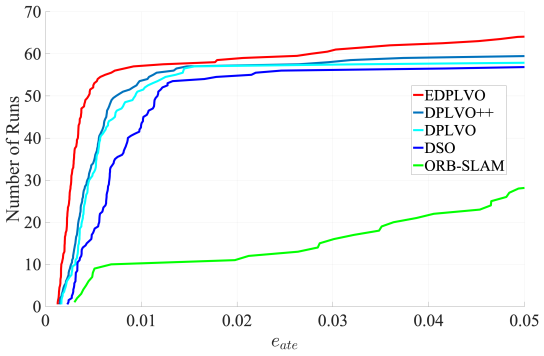


Fig. 6. The cumulative error curves for e_{ate} on the *ICL_NUM* dataset [39]. e_{ate} represents the absolute trajectory error. For an algorithm, a point on its cumulative error curve represents the number of sequences whose absolute trajectory error is smaller than a certain e_{ate}.

VI. Experimental Results

A. Datasets and Metrics

We use the *ICL-NUIM* [39] and *TUM monoVO* [40] datasets to evaluate the performance of the compared algorithms. The *ICL-NUIM* dataset contains 8 indoor sequences, and the *TUM monoVO* dataset has 50 loop-closed sequences from indoor and outdoor environments. To deal with the non-deterministic behavior, we run each sequence 10 times backwards and forwards, and adopt the cumulative error curve to summarize the results, as done in [1]. This curve shows the number of tracked sequences whose errors are below a certain threshold, which reveals both accuracy and robustness of an algorithm. The *ICL-NUIM* dataset provides the ground truth trajectory for each sequence. Thus we compare the absolute trajectory error e_{ate} of the compared algorithm. On the other hand, the *TUM monoVO* dataset does not offer the ground truth trajectory. Instead, it provides the loop-closure ground truth. We adopt the alignment error e_{align}, rotation drift e_r and scale drift e_s defined in [40] to evaluate the performance of an algorithm.

B. Comparison with the State-of-the-art

We compare our algorithm (referred to as **EDPLVO**) with the state-of-the-art visual odometry algorithms, including DSO [1], DPLVO [2] and ORB-SLAM [9]. Fig. 6 and Fig. 7 show the results for the *ICL-NUIM* dataset [39] and the *TUM monoVO* dataset [40], respectively. The results show that our algorithm outperforms other algorithms. The results of ORB-SLAM and DSO are obtained from the website of DSO [1]. Our algorithm is closely related to DPLVO. For a fair comparison, we use the line management approach introduced in Section V to replace the counterpart in DPLVO to make both algorithms have the same front end, and name this method as **DPLVO++**. As shown in Fig. 6, DPLVO++ yields better results than DPLVO. As the two datasets are rich in low-textured environments, ORB-SLAM does not perform well. Due to the proposed line photometric error (10) that imposes exact collinearity into photometric error and the optimization of the full four DoF of the 3D line, our algorithm achieves better accuracy than DPLVO++.

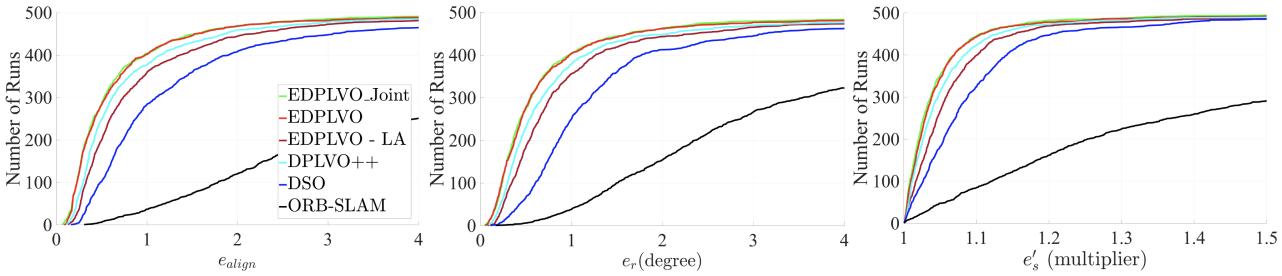


Fig. 7. The cumulative error curves on the *TUM monoVO* [40] dataset. The alignment error e_{align}, rotation drift e_r and scale drift e_s defined in [40] are used to evaluate the performance of the compared algorithms. Here e'_s in the last figure is defined as $e'_s = max(e_s, e_s^{-1})$. A point (e, n) on a cumulative error curve means that there are n tracked sequences whose errors are smaller than e.

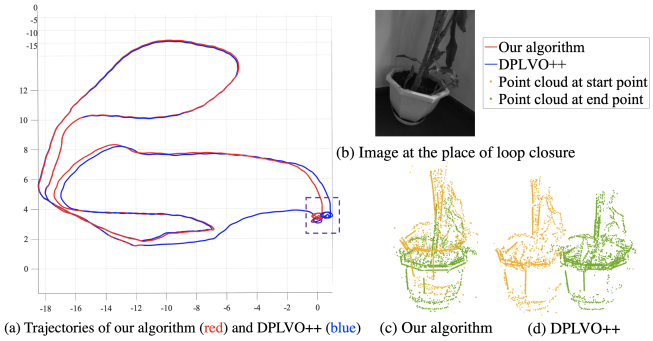


Fig. 8. Accumulated drifts of our algorithm and DPLVO++ [2]. (a) shows the trajectories generated by our algorithm and DPLVO++. (c) and (d) illustrate the point clouds of our algorithm and DPLVO++ at the place of loop closure (b), respectively. The results show that our algorithm yields smaller drift than DPLVO++.

Fig. 1 demonstrates the point cloud and lines generated by our algorithm. Fig. 8 shows the trajectories and the point clouds at the place of loop closure generated by our algorithm and DPLVO++ [2], respectively. The results show that the drift of EDOLVO is smaller than DPLVO++.

C. Ablation Study

We then study the impact of different components of our algorithm. We consider two variants of our algorithm:

- **EDPLVO - LA:** Line association is not established. That is to say the collinearity term in (12) is removed.
- **EDPLVO_Joint:** All variables in cost function (12) are jointly adjusted. We adopt the four DoF representation introduced in [34] to parameterize the 3D line.

Fig. 7 illustrates the results. Due to regulating the collinear points, EDPLVO - LA outperforms DSO. Besides, the performance of EDPLVO - LA approaches DPLVO++ [2]. It is not surprising that EDPLVO outperforms EDPLVO - LA, as the line association introduces additional constraints for poses. The differences between the results of EDPLVO and EDPLVO_Joint are marginal, but EDPLVO is significantly more efficient, as discussed below.

D. Runtime

The main contributions of this paper focus on the back-end. Thus we evaluate the runtime of the back-end of our algorithm, EDPLVO_Joint, DSO [1] and DPLVO++ [2]. The runtime was obtained on a laptop with an i7 3.4 *GHZ* CPU and 16G memory using the *TUM monoVO* dataset. The average runtime of the back-ends of our algorithm, EDPLVO_Joint, DSO [1] and DPLVO++ [2] is about 96*ms*, 123 *ms*, 141 *ms* and 172 *ms* respectively. Our algorithm has the fastest back-end among the compared algorithms. Note that although 3D lines in EDPLVO_Joint are parameterized in four DoF [34] and DPLVO++ adopts a two-DoF parameterization for 3D lines, EDPLVO_Joint is still faster than DPLVO++.

VII. Conclusions

This paper presents a novel direct VO algorithm. We prove that the 3D points of pixels on a 2D line are determined by the inverse depths of the endpoints of the 2D line, which makes incorporating lines into the photometric error feasible. Compared to DPLVO [2], our algorithm significantly decreases the number of variables in the optimization and makes the collinearity exactly met. Furthermore, we introduce a two-step optimization method to speed up the optimization and prove its convergence. The experimental results show that our algorithm significantly reduces the computation load of the optimization and obtains more accurate results than the state-of-the-art VO algorithms.

Appendix

A. Proof of Theorem 1

Proof: The 3D point $\mathbf{p}_{\boldsymbol{x}}$ should be at the intersection of the line $\boldsymbol{L}_{\boldsymbol{l}}$ and the back-projected line $\boldsymbol{L}_{\boldsymbol{x}}$ of $\boldsymbol{x}$, as shown in Fig. 3 (a). $\boldsymbol{L}_{\boldsymbol{x}}$ passes through the origin of the camera. Thus its Plücker coordinates have the form $\boldsymbol{L}_{\boldsymbol{x}} = [\mathbf{0}; \boldsymbol{d}_{\boldsymbol{x}}]$, where $\boldsymbol{d}_{\boldsymbol{x}}$ is defined in (7). According to (4), $\mathbf{p}_{\boldsymbol{x}}$ should satisfy the following equation system

$$\begin{aligned} \boldsymbol{m}_{\boldsymbol{l}} - \mathbf{p}_{\boldsymbol{x}} \times \boldsymbol{d}_{\boldsymbol{l}} &= \mathbf{0}_{3\times 1}, \\ \mathbf{p}_{\boldsymbol{x}} \times \boldsymbol{d}_{\boldsymbol{x}} &= \mathbf{0}_{3\times 1}. \end{aligned} \tag{13}$$

As $\mathbf{p}_{\boldsymbol{x}} \times \boldsymbol{d}_{\boldsymbol{l}} = -[\boldsymbol{d}_{\boldsymbol{l}}]_\times \mathbf{p}_{\boldsymbol{x}}$ and $\mathbf{p}_{\boldsymbol{x}} \times \boldsymbol{d}_{\boldsymbol{x}} = -[\boldsymbol{d}_{\boldsymbol{x}}]_\times \mathbf{p}_{\boldsymbol{x}}$, the linear system (13) can be rewritten as

$$\begin{aligned} [\boldsymbol{d}_{\boldsymbol{l}}]_\times \mathbf{p}_{\boldsymbol{x}} &= -\boldsymbol{m}_{\boldsymbol{l}}, \\ [\boldsymbol{d}_{\boldsymbol{x}}]_\times \mathbf{p}_{\boldsymbol{x}} &= \mathbf{0}_{3\times 1}. \end{aligned} \tag{14}$$

Using the definition of $\mathbf{A}_{\boldsymbol{l}}$ and $\boldsymbol{b}_{\boldsymbol{l}}$ in (8), the equation system (14) can be written as $\mathbf{A}_{\boldsymbol{l}}\mathbf{p}_{\boldsymbol{x}} = \boldsymbol{b}_{\boldsymbol{l}}$. Thus $\mathbf{p}_{\boldsymbol{x}}$ has a closed-form solution $\mathbf{p}_{\boldsymbol{x}} = \left(\mathbf{A}_{\boldsymbol{l}}^T \mathbf{A}_{\boldsymbol{l}}\right)^{-1} \mathbf{A}_{\boldsymbol{l}}^T \boldsymbol{b}_{\boldsymbol{l}}$. As the elements of $\mathbf{A}_{\boldsymbol{l}}$ and $\boldsymbol{b}_{\boldsymbol{l}}$ only depend on d_1 and d_2, $\mathbf{p}_{\boldsymbol{x}}$ is determined by d_1 and d_2. ■

B. Proof of Theorem 2

Proof: Let us denote as $\boldsymbol{E}_k$ the value of cost (12) at the kth iteration. At the $(k+1)$th iteration, we first minimize the last term $E_{\boldsymbol{L}}$ in (12) with the fixed poses and inverse depths obtained at the kth iteration. Assume the value of (12) after this step is $E_{k+1}^{[1]}$. As the first two terms in (12) do not involve the 3D lines, they keep the same value after this step. Thus it is clear that $E_{k+1}^{[1]} \leq E_k$. In the second step, we fix the value of the 3D lines and conduct one LM step for the poses and inverse depths to reduce the value of $E_{k+1}^{[1]}$. Assume the value of (12) after this step is E_{k+1}. It is obvious that

$$0 \leq E_{k+1} \leq E_{k+1}^{[1]} \leq E_k. \tag{15}$$

According to the monotone convergence theorem (*i.e.*, if a sequence is decreasing and bounded below, this sequence has a limit), the two-step minimization algorithm always converges.

■

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] L. Zhou, S. Wang, and M. Kaess, "DPLVO: Direct Point-Line Monocular Visual Odometry," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7113–7120, 2021.

[3] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[4] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[5] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7286–7291.

[6] L. Zhou and M. Kaess, "Windowed bundle adjustment framework for unsupervised learning of monocular depth estimation with u-net extension and clip loss," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3283–3290, 2020.

[7] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.

[8] S. Li, X. Wu, Y. Cao, and H. Zha, "Generalizing to the open world: Deep visual odometry with online adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 184–13 193.

[9] 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.

[10] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981.

[11] 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.

[12] L. von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization," in *International Conference on Robotics and Automation (ICRA)*, May 2018.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Q. Wang, Z. Yan, J. Wang, F. Xue, W. Ma, and H. Zha, "Line Flow Based Simultaneous Localization and Mapping," *IEEE Transactions on Robotics*, pp. 1–17, 2021.

[18] 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.

[19] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 626–635.

[20] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 962–971.

[21] Z. Zhang, Z. Li, N. Bi, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, "Ppgnet: Learning point-pair graph for line segment detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7105–7114.

[22] N. Xue, T. Wu, S. Bai, F. Wang, G.-S. Xia, L. Zhang, and P. H. Torr, "Holistically-attracted wireframe parsing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2788–2797.

[23] K. Zhao, Q. Han, C.-B. Zhang, J. Xu, and M.-M. Cheng, "Deep hough transform for semantic line detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[24] 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.

[25] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[26] 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 multimap slam," *IEEE Transactions on Robotics*, 2021.

[27] 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.

[28] 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.

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

[30] 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.

[31] M. Gladkova, R. Wang, N. Zeller, and D. Cremers, "Tight Integration of Feature-based Relocalization in Monocular Direct Visual Odometry," in *2021 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2021.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: Incremental, Consistent and Efficient Bundle Adjustment for Visual-Inertial SLAM," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

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

[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] J. Engel, V. Usenko, and D. Cremers, "A Photometrically Calibrated Benchmark For Monocular Visual Odometry," in *arXiv:1607.02555*, July 2016.