2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Robust Visual SLAM with Point and Line Features

Xingxing Zuo[1], Xiaojia Xie[1], Yong Liu[1,2] and Guoquan Huang[3]

Abstract— **In this paper, we develop a robust efficient visual SLAM system that utilizes heterogeneous point and line features. By leveraging ORB-SLAM [1], the proposed system consists of stereo matching, frame tracking, local mapping, loop detection, and bundle adjustment of both point and line features. In particular, as the main theoretical contributions of this paper, we, for the first time, employ the orthonormal representation as the minimal parameterization to model line features along with point features in visual SLAM and analytically derive the Jacobians of the re-projection errors with respect to the line parameters, which significantly improves the SLAM solution. The proposed SLAM has been extensively tested in both synthetic and real-world experiments whose results demonstrate that the proposed system outperforms the state-of-the-art methods in various scenarios.**

I. Introduction

Visual SLAM (V-SLAM) is one of enabling technologies for autonomous systems such as self-driving cars, unmanned aerial vehicles and space robots. While most V-SLAM solutions rely on point features due to their simplicity, line features commonly seen in man-made environments are less sensitive to lighting variation and position ambiguity and have been only used in recent work [2]–[8]. In principle, the combination of point and line features would provide more geometric constraints about the structure of the environment than either one, which motivates us to design robust V-SLAM with point and line features.

Recently, optimization-based approaches have become favorable for the V-SLAM due to its superior accuracy per computational unit as compared with filtering-based approaches [9]. In particular, graph-based SLAM is one of the most popular formulations which constructs a factor graph whose nodes correspond to the states to estimate and edges represent measurement constraints between the nodes. When incorporating the line features into the traditional point feature-based graph SLAM framework, two challenges arise: The first one is that the spatial line is often over parameterized for the convenience of transformation [3], [4], [7], which incurs extra computational overhead in the graph optimization. Note that while a spatial line has only *four* degrees of freedom, typically it is represented by its two spatial endpoints or the $Plücker$ coordinates with *six* degrees of freedom. Secondly, it is known that the Jacobian plays an important role when using an iterative approach to solve the graph optimization problem. In part because of the over parametrization, most approaches [6], [7] using line features typically employ the numerically computed Jacobians, which incurs approximation. In contrast, we analytically compute the Jacobians during the graph optimization in order to improve accuracy as well as efficiency.

In particular, this paper introduces a robust and efficient graph-based visual SLAM system using both point and line features with a unified cost function, combining the re-projection errors of points and lines. In our back-end, the spatial line is parametrized by the orthonormal representation, which is the minimal and decoupled representation. Based on this minimal parametrization, we further derive the analytical Jacobian of the line re-projection error. Specifically, the main contributions of this paper are the following:

- An improved extraction and matching method for line features is introduced to robustify data association.
- In the back-end of the proposed visual SLAM, we employ the orthonormal (minimal) representation to parameterize lines and analytically compute the corresponding Jacobians.
- We design and implement a complete visual SLAM system using both point and line features, which includes stereo matching, frame tracking, local mapping, bundle adjustment of both line feature and point feature, as well as point-line based loop detection. Extensive experimental results are presented to validate its performance.

II. Related Work

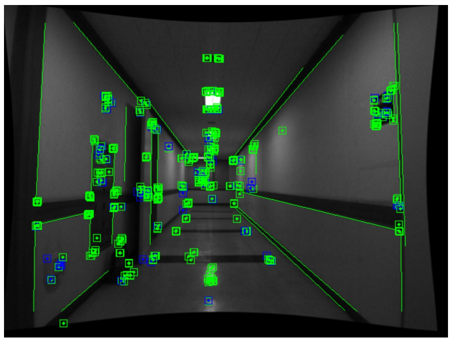
Some methods have been proposed to parameterize line in three-dimensional (3D) space efficiently. Sola [10] summarizes several methods to represent line including $Plücker$ coordinates, Anchored $Plücker$ Lines, and homogeneous-points line etc. For minimizing the number of the parameters, Bartoli [11] proposed the orthonormal representation with the minimum four parameters to represent spatial lines in SFM.

Combination of point and line features has been utilized in the SLAM community recently. Marzorati et al. [3] proposed a SLAM with points and lines, which uses a special trifocal cameras to detect and reconstruct 3D lines. Rother [2] reconstructed points and lines at the cost of requiring a reference plane to be visible in all views. Koletschka et al. [5] proposed a stereo odometry based on points and lines, which computes the sub-pixel disparity of the endpoints of the line and deals with partial occlusions. Lu [7] fuses point and line features to form a RGBD visual odometry algorithm, which extracts 3D points and lines from RGB-D data. It is also proved that fusing these two types of features

[1]Xingxing Zuo and Xiaojia Xie are with the Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China.

[2]Yong Liu is with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China (Yong Liu is the corresponding author, email: yongliu@iipc.zju.edu.cn).

[3]Guoquan Huang is with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716, USA.


(a) Point and line features detected in one image

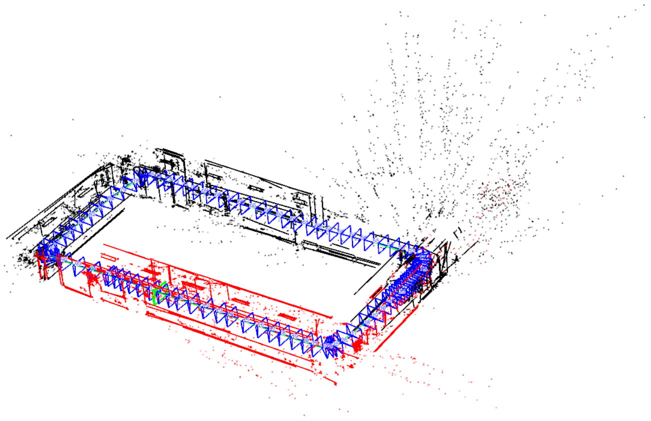

(b) The point and line map

Fig. 1. The proposed visual SLAM with point and line features on the it3f dataset [6]. Note that in (b), the green lines indicate the trajectory of camera motion. The blue frames represent keyframes, the current frame in green, and the local map for the tracking at that moment in red.

produced smaller uncertainty in motion estimation than using either feature type alone in his work. Ruben [8] proposed a probabilistic approach to stereo visual odometry based on the combination of points and line segments, which weighs the associated errors of points and line segments according to their covariance matrices.

III. Detection and Representation of Line Features

A. Extraction and Description of Line Features

Line Segment Detector (LSD) [12] is a popular feature detector for line segments. It is designed to work on noisy image in various scenes without parameter tuning and is able to provide subpixel accuracy. However, the LSD suffers from the problem of dividing a line into multiple segments in some scenarios as shown in Fig. 2, causing failures for matching and tracking line features.

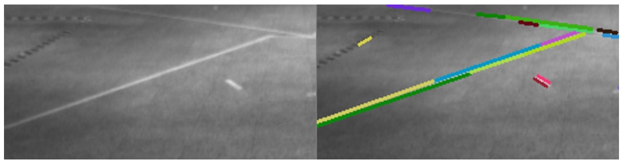


Fig. 2. Performance of LSD. Left: Original image. Right: Line features detected in the image by LSD.

Therefore, in this paper, we seek to improve the LSD algorithm by minimizing the influence of dividing a line into multiple segments. In particular, we merge the line segments which should be on the same one straight line while being divided to several parts. For each line segment extracted by LSD, the start point and end point can be distinguished, because the direction is encoded by which side of the line segment is darker. In our improvement, we merge the segments according to their differences of both direction and distance. As shown in Fig. 3, l represents the minimum distance between the endpoints of the two segments, and d indicates the distance from the midpoint of one segment to the other line segment. If d and l are smaller than the given threshold and the direction difference is also small, then the two segments are considered to be the candidates to be combined. This improved line detector has the advantages of making more robust and accurate data association as demonstrated in our experiments. Fig. 4 shows the result of the two different detectors. Note that the merged line segments found by our improved detector is represented by the LBD line descriptor [13], which is a 256-bit vector same as ORB point descriptor [14]. The distance between the two descriptors can be another criterion for fusing two lines.

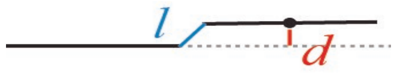


Fig. 3. Distances between two line segments.

B. Line Feature Matching

Based on the LBD line segment descriptor, we introduce the geometric properties [15] of line to perform effective line matching. In our approach, two successfully matched line features $\boldsymbol{l}_1$, $\boldsymbol{l}_2$ need to satisfy the following conditions:

1) the angular difference of two matched lines is smaller than a given threshold Φ;
2) the length of $\boldsymbol{l}_1$, $\|\boldsymbol{l}_1\|$ is similar to the length of $\boldsymbol{l}_2$, $\|\boldsymbol{l}_2\|$: $\frac{min(\|\boldsymbol{l}_1\|,\|\boldsymbol{l}_2\|)}{max(\|\boldsymbol{l}_1\|,\|\boldsymbol{l}_2\|)} > \tau$;
3) the overlapping length of the two lines is greater than a certain value: $\frac{l_{overlap}}{\min(\|\boldsymbol{l}_1\|,\|\boldsymbol{l}_2\|)} > \beta$;

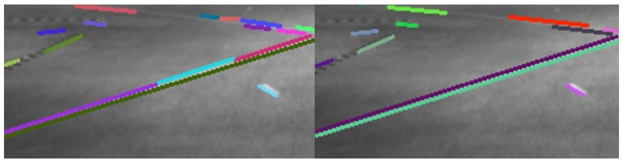


Fig. 4. Comparative results of different detectors. Left: The original LSD detector. Right: The proposed improved detector.

4) the distance between the two LBD descriptors is less than a certain value.

C. Geometric Representation

As 3D line can be initialized by its two spatial points, we assume that their homogeneous coordinates are $\boldsymbol{X}_1 = (x_1, y_1, z_1, r_1)^T$, $\boldsymbol{X}_2 = (x_2, y_2, z_2, r_2)^T$ respectively, while the inhomogeneous coordinates are represented as $\tilde{\boldsymbol{X}}_1$, $\tilde{\boldsymbol{X}}_2$. Then *Plücker* coordinates can be constructed as follows:

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} \tilde{\boldsymbol{X}}_1 \times \tilde{\boldsymbol{X}}_2 \\ r_2\tilde{\boldsymbol{X}}_1 - r_1\tilde{\boldsymbol{X}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} \\ \boldsymbol{v} \end{bmatrix} \in \mathbb{P}^5 \subset \mathbb{R}^6 \quad (1)$$

which is a 6-dimensional vector consisting of $\boldsymbol{n}$ and $\boldsymbol{v}$. $\boldsymbol{v}$ is the direction vector of the line and $\boldsymbol{n}$ is the normal vector of the plane determined by the line and the origin [10].

Since 3D line has only four degrees of freedom, the *Plücker* coordinates are over parameterized. In the back-end graph optimization, the extra degrees of freedom will increase the computational cost and cause the numerical instability of the system. Thus, Bartoli [11] proposed the orthonormal representation with minimum four parameters. We can obtain the orthonormal representation $(\boldsymbol{U}, \boldsymbol{W}) \in SO(3) \times SO(2)$ from *Plücker* coordinates:

$$\boldsymbol{\mathcal{L}} = [\boldsymbol{n}|\boldsymbol{v}] = \begin{bmatrix} \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|} & \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} & \frac{\boldsymbol{n}\times\boldsymbol{v}}{\|\boldsymbol{n}\times\boldsymbol{v}\|} \end{bmatrix} \begin{bmatrix} \|\boldsymbol{n}\| & 0 \\ 0 & \|\boldsymbol{v}\| \\ 0 & 0 \end{bmatrix} \quad (2)$$

$$= \boldsymbol{U} \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \\ 0 & 0 \end{bmatrix}. \quad (3)$$

The orthonormal representation of line $(\boldsymbol{U}, \boldsymbol{W})$ consists of:

$$\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{\theta}) = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \quad (4)$$

$$\boldsymbol{W} = \boldsymbol{W}(\theta) = \begin{bmatrix} w_1 & -w_2 \\ w_2 & w_1 \end{bmatrix}. \quad (5)$$

We can update the orthonormal representation with the minimum four parameters $\boldsymbol{\delta}_\theta = [\boldsymbol{\theta}^T, \theta]^T \in \mathbb{R}^4$, we can update $\boldsymbol{U}$ with the vector $\boldsymbol{\theta} \in \mathbb{R}^3$, and update $\boldsymbol{W}$ with θ. Each sub-parameter of $\boldsymbol{\delta}_\theta$ has a specific geometric interpretation. $\boldsymbol{W}$ updated by θ encapsulates the vertical distance $\mathbf{d}$ from the origin to the spatial line. As shown in Fig. 5, in the case of fixed $\boldsymbol{W}$ represented in gray, the three-dimensional vector $\boldsymbol{\theta}$ is related to the rotation of the line around three axes, drawn in orange, green, and blue.

Note that in the proposed visual SLAM system, we only use the orthonormal representation in the back-end optimization, as it is the minimal and decoupled representation. However, in the other steps, the *Plücker* coordinates are used due to its convenience in camera projection, endpoints trimming, and line initialization [6], [11].

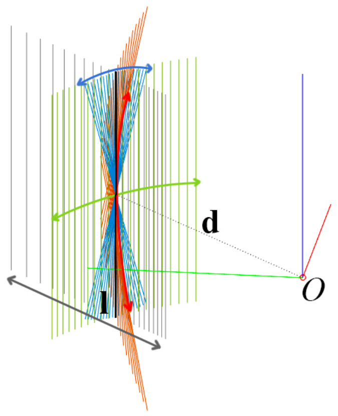


Fig. 5. Geometric interpretation of four parameters $\boldsymbol{\delta}_\theta$ in updating orthonormal representation.

IV. Graph Optimization with Point and Line Measurements

In what follows, we present in detail how the line measurements are incorporated into our graph-based visual SLAM system, while the point measurements are treated in a standard way, for example, as in ORB-SLAM [1].

A. Measurement Models of Point and Line Features

We use the transformation matrix $\boldsymbol{T}_{cw} \in SE(3)$ to denote the transformation from world frame to camera frame, which consists of a rotation matrix $\boldsymbol{R}_{cw} \in SO(3)$, and a translation vector $\boldsymbol{t}_{cw} \in \mathbb{R}^3$, as shown in (6). First, we convert the 3D line $\boldsymbol{\mathcal{L}}_w$ from the world frame to the camera frame [16] as shown in (7), denoted as $\boldsymbol{\mathcal{L}}_c$, with representation of the *Plücker* coordinates. Then the 3D line $\boldsymbol{\mathcal{L}}_c$ is projected to the image in (8), described as $\boldsymbol{l}'$ on image plane, according to the known intrinsic parameters of camera. It should be noted that only normal components $\boldsymbol{n}_c$ in the *Plücker* coordinates $\boldsymbol{\mathcal{L}}_c$ can provide meaningful information in the projection. Then, the re-projection error of 3D line is represented as the distance between two homogeneous endpoints $\boldsymbol{x}_s$, $\boldsymbol{x}_e$ of the matched line segment $\boldsymbol{z}$ to the back-projected line $\boldsymbol{l}'$ on image plane as shown in (9).

$$\boldsymbol{T}_{cw} = \begin{bmatrix} \boldsymbol{R}_{cw} & \boldsymbol{t}_{cw} \\ \boldsymbol{0} & 1 \end{bmatrix} \quad (6)$$

$$\boldsymbol{\mathcal{L}}_c = \begin{bmatrix} \boldsymbol{n}_c \\ \boldsymbol{v}_c \end{bmatrix} = \boldsymbol{\mathcal{H}}_{cw}\boldsymbol{\mathcal{L}}_w = \begin{bmatrix} \boldsymbol{R}_{cw} & [\boldsymbol{t}_{cw}]_\times \boldsymbol{R}_{cw} \\ \boldsymbol{0} & \boldsymbol{R}_{cw} \end{bmatrix} \boldsymbol{\mathcal{L}}_w, \quad (7)$$

where $[.]_\times$ denotes the skew-symmetric matrix of a vector, and $\boldsymbol{\mathcal{H}}_{cw}$ represents transformation matrix of the line.

$$\boldsymbol{l}' = \boldsymbol{\mathcal{K}}\boldsymbol{n}_c = \begin{bmatrix} f_v & 0 & 0 \\ 0 & f_u & 0 \\ -f_v c_u & f_u c_v & f_u f_v \end{bmatrix} \boldsymbol{n}_c = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix}, \quad (8)$$

where $\boldsymbol{\mathcal{K}}$ denotes the projection matrix of the line [10].

$$e_l = \mathrm{d}(\boldsymbol{z}, \boldsymbol{l}') = \left[\frac{\boldsymbol{x}_s^{\mathrm{T}}\boldsymbol{l}'}{\sqrt{l_1^2 + l_2^2}}, \frac{\boldsymbol{x}_e^{\mathrm{T}}\boldsymbol{l}'}{\sqrt{l_1^2 + l_2^2}} \right]^{\mathrm{T}}, \quad (9)$$

where $\mathrm{d}(.)$ denotes the distance function.

The camera pose $\boldsymbol{T}_{kw}$, the 3D point position $\boldsymbol{X}_{w,i}$, and the position of 3D line $\boldsymbol{\mathcal{L}}_{w,j}$ are denoted as vertices in the graph model. The two types of edge, the pose-point edge in (10), the pose-line edge in (11), are constructed according to the front-end data association. The re-projection errors encapsulated in the edges are:

$$Ep_{k,i}=\boldsymbol{x}_{k,i}-\mathbf{K}\boldsymbol{T}_{kw}\boldsymbol{X}_{w,i} \tag{10}$$

$$El_{k,j}=\mathrm{d}\left(\boldsymbol{z}_{k,j},\boldsymbol{\mathcal{K}}\mathrm{n_c}[\boldsymbol{\mathcal{H}}_{\mathrm{cw}}\boldsymbol{\mathcal{L}}_{\mathrm{w,j}}]\right), \tag{11}$$

where $\boldsymbol{x}_{k,i}$ stands for the coordinates of point in the image, $\mathrm{n_c}[.]$ denotes the normal components of the *Plücker* coordinates. For simplicity, we omit the conversion from homogeneous coordinates to the inhomogeneous in the above equations. Assuming that the observations obey Gaussian distribution, the final cost function C can be obtained as in (12), Where $\boldsymbol{\Sigma p}^{-1}$, $\boldsymbol{\Sigma l}^{-1}$ are the inverse covariance matrices of points and lines, and ρ_p, ρ_l are robust Huber cost functions. The back-end optimization minimizes the cost function C.

$$C=\sum_{k,i}\rho_p\left(Ep_{k,i}^{\mathrm{T}}\boldsymbol{\Sigma p}_{k,i}^{-1}Ep_{k,i}\right)+\sum_{k,j}\rho_l\left(El_{k,j}^{\mathrm{T}}\boldsymbol{\Sigma l}_{k,j}^{-1}El_{k,j}\right) \tag{12}$$

B. Jacobian of Line Re-projection Error

It is known that the Jacobian is important when using an iterative approach to solve the the graph optimization problem. To the best of our knowledge, this is the first paper deriving out the analytical Jacobains of re-projection errors with respect to line parameters, which including the Jacobian with respect to the small pose changes $\boldsymbol{\delta}_\xi$, and to the four dimensional vector $\boldsymbol{\delta}_\theta$ which updates the orthonormal representation. The Jacobian of the line re-projection error $el=\mathrm{d}(\boldsymbol{z},\boldsymbol{l}')$ with respect to the back-projected line $\boldsymbol{l}'=[l_1,l_2,l_3]^T$ is given by:

$$\frac{\partial el}{\partial \boldsymbol{l}'}=\frac{1}{l_n}\begin{bmatrix} u_1-\frac{l_1e_1}{l_n^2} & v_1-\frac{l_2e_1}{l_n^2} & 1 \\ u_2-\frac{l_1e_2}{l_n^2} & v_1-\frac{l_2e_2}{l_n^2} & 1\end{bmatrix}_{2\times 3}, \tag{13}$$

where $e_1=\boldsymbol{x}_s^{\mathrm{T}}\boldsymbol{l}'$, $e_2=\boldsymbol{x}_e^{\mathrm{T}}\boldsymbol{l}'$, $l_n=\sqrt{(l_1^2+l_2^2)}$. $\boldsymbol{x}_s=[u_1,v_1,1]^{\mathrm{T}}$ and $\boldsymbol{x}_e=[u_2,v_2,1]^{\mathrm{T}}$ are the two endpoints of matched line segment in the image.

Recall the projection of 3D line $\boldsymbol{l}'=\boldsymbol{\mathcal{K}}\boldsymbol{n}_c$, then:

$$\frac{\partial \boldsymbol{l}'}{\partial \boldsymbol{\mathcal{L}}_c}=\frac{\partial \boldsymbol{\mathcal{K}}\boldsymbol{n}_c}{\partial \boldsymbol{\mathcal{L}}_c}=\begin{bmatrix}\boldsymbol{\mathcal{K}} & \mathbf{0}\end{bmatrix}_{3\times 6} \tag{14}$$

Assuming that the orthonormal representation of line in the world frame $\boldsymbol{\mathcal{L}}_w$, which consists of $\boldsymbol{U}$ and $\boldsymbol{W}$, We write the Jacobians directly:

$$\frac{\partial \boldsymbol{\mathcal{L}}_c}{\partial \boldsymbol{\mathcal{L}}_w}=\frac{\partial \boldsymbol{\mathcal{H}}_{cw}\boldsymbol{\mathcal{L}}_w}{\partial \boldsymbol{\mathcal{L}}_w}=\boldsymbol{\mathcal{H}}_{cw} \tag{15}$$

$$\frac{\partial \boldsymbol{\mathcal{L}}_w}{\partial \boldsymbol{\delta}_\theta}=\begin{bmatrix}-[w_1\boldsymbol{u}_1]_\times & -w_2\boldsymbol{u}_1 \\ -[w_2\boldsymbol{u}_2]_\times & w_1\boldsymbol{u}_2\end{bmatrix}_{6\times 4}, \tag{16}$$

where $\boldsymbol{u}_i$ is the i_{th} column of $\boldsymbol{U}$.

It is difficult to compute $\frac{\partial \boldsymbol{\mathcal{L}}_w}{\partial \boldsymbol{\delta}_\xi}$ directly, so we divide the pose changes $\boldsymbol{\delta}_\xi$ into two parts, the translation part $\boldsymbol{\delta}_\rho$ and the rotation part $\boldsymbol{\delta}_\phi$. $\boldsymbol{\delta}_\phi$ are set to zeros when computing Jacobian with respect to $\boldsymbol{\delta}_\rho$. With a transformation matrix $\boldsymbol{T}^*$ containing the translation $\boldsymbol{\delta}_\rho$, the new line $\boldsymbol{\mathcal{L}}_c^*$ is:

$$\boldsymbol{T}^*=\exp\left(\boldsymbol{\delta}_\xi{}^\wedge\right)T_{cw}\approx\begin{bmatrix}\boldsymbol{I} & \boldsymbol{\delta}_\rho \\ \mathbf{0}^T & 1\end{bmatrix}T_{cw} \tag{17}$$

$$\boldsymbol{R}^*=\boldsymbol{R}_{cw},\ \boldsymbol{t}^*=\boldsymbol{\delta}_\rho+\boldsymbol{t}_{cw} \tag{18}$$

$$\boldsymbol{\mathcal{H}}_{cw}^*=\begin{bmatrix}\boldsymbol{R}_{cw} & [\boldsymbol{\delta}_\rho+\boldsymbol{t}_{cw}]_\times\boldsymbol{R}_{cw} \\ \mathbf{0} & \boldsymbol{R}_{cw}\end{bmatrix} \tag{19}$$

$$\boldsymbol{\mathcal{L}}_c^*=\boldsymbol{\mathcal{H}}_{cw}^*\boldsymbol{\mathcal{L}}_w=\begin{bmatrix}\boldsymbol{R}_{cw}\boldsymbol{n}_w+[\boldsymbol{\delta}_\rho+\boldsymbol{t}_{cw}]_\times\boldsymbol{R}_{cw}\boldsymbol{v}_w \\ \boldsymbol{R}_{cw}^T\boldsymbol{v}_w\end{bmatrix}, \tag{20}$$

where $\exp\left(\boldsymbol{\delta}_\xi{}^\wedge\right)$ denotes the exponential map from Lie algebras to Lie Groups (hence $\boldsymbol{\delta}_\xi{}^\wedge$ is a Lie algebra). Then it is easy to deduce the partial derivative of $\boldsymbol{\delta}_\rho$:

$$\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\rho}=\begin{bmatrix}\frac{[\boldsymbol{\delta}_\rho+\boldsymbol{t}_{cw}]_\times\boldsymbol{R}_{cw}\boldsymbol{v}_w}{\partial\boldsymbol{\delta}_\rho} \\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}-[\boldsymbol{R}_{cw}\boldsymbol{v}_w]_\times \\ \mathbf{0}\end{bmatrix}_{6\times 3} \tag{21}$$

The process to deduce $\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\phi}$ is similar to $\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\rho}$, except for $\boldsymbol{\delta}_\rho=\mathbf{0}$. We only shows the final result Eq.22, and drops the coordinate frame subscripts for readability. Readers can refer to the Appendix for more details.

$$\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\phi}==\begin{bmatrix}-[\boldsymbol{Rn}]_\times-\left[[\boldsymbol{t}]_\times\boldsymbol{Rv}\right]_\times \\ -[\boldsymbol{Rv}]_\times\end{bmatrix}_{6\times 3} \tag{22}$$

Stacking the Jacobians of $\boldsymbol{\delta}_\rho$ and $\boldsymbol{\delta}_\phi$, we can obtain the final Jacobian of $\boldsymbol{\delta}_\xi$:

$$\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\xi}=\begin{bmatrix}-[\boldsymbol{Rn}]_\times-\left[[\boldsymbol{t}]_\times\boldsymbol{Rv}\right]_\times & -[\boldsymbol{Rv}]_\times \\ -[\boldsymbol{Rv}]_\times & \mathbf{0}\end{bmatrix}_{6\times 6} \tag{23}$$

Finally, the Jacobian of the re-projection error with respect to the line parameters can be found using the chain rule:

$$Jl_\xi=\frac{\partial e_l}{\partial \boldsymbol{\delta}_\xi}=\frac{\partial e_l}{\partial \boldsymbol{l}'}\frac{\partial \boldsymbol{l}'}{\partial \boldsymbol{\mathcal{L}}_c}\frac{\partial \boldsymbol{\mathcal{L}}_c}{\partial \boldsymbol{\delta}_\xi} \tag{24}$$

$$Jl_\theta=\frac{\partial e_l}{\partial \boldsymbol{\delta}_\theta}=\frac{\partial e_l}{\partial \boldsymbol{l}'}\frac{\partial \boldsymbol{l}'}{\partial \boldsymbol{\mathcal{L}}_c}\frac{\partial \boldsymbol{\mathcal{L}}_c}{\partial \boldsymbol{\mathcal{L}}_w}\frac{\partial \boldsymbol{\mathcal{L}}_w}{\partial \boldsymbol{\delta}_\theta} \tag{25}$$

Once these analytical Jacobians are available, we can employ iterative algorithms such as Gaussian-Newton to solve the graph optimization problem.

V. Experimental Results

A. System Implementation

The proposed visual SLAM system is designed and implemented based on ORB-SLAM2 [1] and has three main parallel threads (see Fig. 6): Tracking, Local Mapping and Loop Closing. The global BA thread is started only after finishing loop closing. In the following, we briefly describe each component while focusing on the difference from [1].

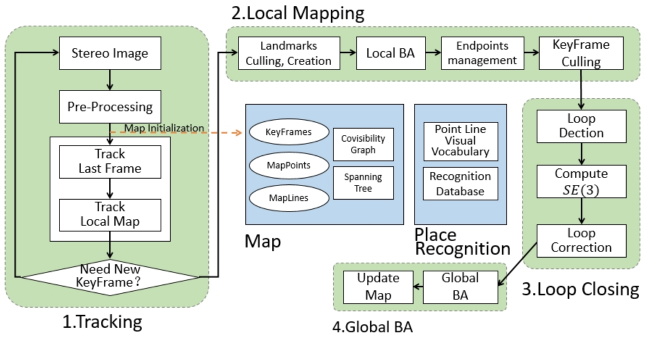


Fig. 6. The architecture of the proposed graph-based visual SLAM system using both point and line features.

1) Tracking: Our system uses rectified stereo image sequence as input. For every input frame, four threads are launched to extract point feature (Keypoints) and line feature (Keylines) for left and right image in parallel. ORB features are applied for point feature detection and description. Line feature is detected by LSD and described by LBD descriptor. Then two threads are launched for stereo matching and all the features are classified as stereo or monocular features according to whether the feature in the left image could find its stereo match in the right image, as shown in Fig. 7. The stereo matching of 3D lines performs as described in Section III-B. For each monocular feature, we search a match with other unmatched features in other keyframes. Once finding the matched feature, we triangulate the feature in the same way as stereo features.

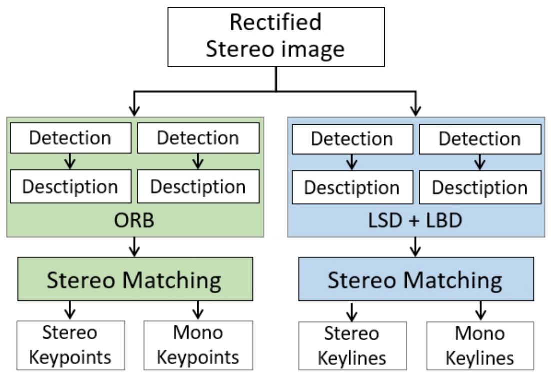


Fig. 7. The workflow of pre-processing images.

Motion estimation is made by two types of tracking, namely tracking last frame and tracking local map. The former one gives an initial pose estimation with the correspondences of adjacent frame, while the latter one refines the pose with much more constraints between the current frame and the local map. After the data association, the pose is estimated by motion-only BA using Levenberg-Marquardt algorithm [17].

We use a constant velocity motion model to predict the camera pose as a prior when tracking last frame. Once the prior is known, the map points and the map lines in the last frame or the local map can be projected to current frame to build more associations. Then we perform a guided search to bound the complexity and enhance the accuracy of matching. Since the 3D line may be partially observed, the projected 2D line cannot be handled the same as the projected 2D point. Fig. 8 shows a simple example, the dash lines can't be observed by the camera while the solid lines can. In order to ensure the visibility of the projected 2D line segments in image plane, we propose a culling based method described as follow:

1) Transform the 3D line $\mathcal{L}_w$ from world frame to current frame according to the prior $\boldsymbol{T}_{kw}'$. Compute the two endpoints $\boldsymbol{X}_{sk}$ and $\boldsymbol{X}_{ek}$.
2) Discard the line if both $\boldsymbol{X}_{sk}$ and $\boldsymbol{X}_{ek}$ are behind the camera. If one of them is behind the camera, compute the intersection of the plane and the 3D line by $\boldsymbol{X}_{ik} = \boldsymbol{X}_{sk} + \lambda\left(\boldsymbol{X}_{sk} - \boldsymbol{X}_{ek}\right)$, where λ is a value between 0 and 1, as depicted in Fig. 8.
3) Project the two 3D endpoints in front of the camera to image plane. Since the projected line maybe lays across or even out of the image bound, all the projected lines must be dealt by Liang-Barsky algorithm [18] which is an efficient line clipping algorithm and can retain the orientation of original line.

Then line matching can be done efficiently thanks to the restricted searching space and binary descriptor. The last step is to decide whether the current frame is a new keyframe. We adopt the same policy as ORB-SLAM2 [1] and add more conditions related to line features.

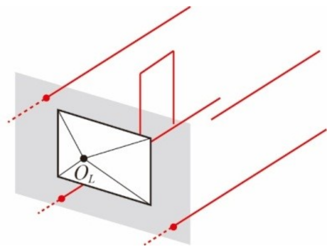


Fig. 8. Partial observation of 3D line. (The dash lines can't be observed by the camera while the solid lines can. The red points denotes the intersection of the plane and the 3D line.)

2) Local Mapping: Once a new keyframe is added, the connections between the current keyframe and other frames will be updated by providing the co-visible information. Local mapping triangulates more map points and lines, removes outlier landmarks, and deletes redundant keyframe. All the camera poses and landmarks in the local map are adjusted by performing local BA. During back-end optimization, the 3D line is parameterized by infinite spatial line, hence its endpoints have no affect on the final optimization results. However, the endpoints play an important role in matching and visualizing, so our system need to maintain two endpoints of the 3D line after optimization. It can be done by back-projecting the 2D line in current keyframe and trimming the corresponding 3D line, which is similar to SLSLAM [6].

3) Loop Closing and Global BA: The loop closing thread is used to reduce drift accumulated during exploration by loop detection and loop correction. Loop detection try to find candidate keyframes based on the technique of bags of words. The visual vocabulary should be trained offline with both point and line features. Here we cluster the ORB features and LBD features to build their own vocabulary by DBOW [19] respectively. For every input keyframe, it is converted to the bag of words vector and stored in the online database. The similarity score between two bag of vector $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ can be computed as follow:

$$\mathrm{s} = \lambda \mathrm{s_p}(\boldsymbol{v}_a, \boldsymbol{v}_b) + (1-\lambda)\,\mathrm{s_l}(\boldsymbol{v}_a, \boldsymbol{v}_b), \tag{26}$$

where λ is an empirical weight coefficient related to scenes. $\mathrm{s_p}(\boldsymbol{v}_a, \boldsymbol{v}_b)$ and $\mathrm{s_l}(\boldsymbol{v}_a, \boldsymbol{v}_b)$ are the similarity score of point feature and line feature. Then we can find the correspondences between the new keyframe and the candidate keyframe. we also refine the correspondences with time consistency test [20]. And try to compute a $SE(3)$ transformation matrix by EPnP [21] with corresponding points in a RANSAC scheme [22]. If failed, we alternatively compute a $SE(3)$ by a method proposed in [23] using the matching lines across two stereo views. Finally, a pose graph optimization is launched to correct the loop. Once finished, a global BA is incorporated to achieve the optimal solution in a separate thread.

B. Results

Various experiments have been conducted in both synthetic and real-world scene. The accuracy and time efficiency of our approach are analyzed. In these experiments, the algorithm run on a computer with Intel Core i7-2600 @ 3.40GHz and 16GB memory in a 64-bit Linux operating system.

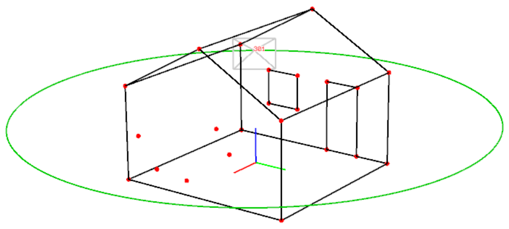


Fig. 9. Synthetic scene with 25 lines and variable number of points.

1) Synthetic data: There is an accurate data association in synthetic Scene. And this experiment is proposed to verify the correctness and advantage of the introduced line feature in the back-end optimization. The derived Jacobian of 3D line re-projection error is used in the optimization. The synthetic scene in Fig. 9 contains a house with a total of 25 lines and variable number of points. This construction is similar to the scene in [10]. Virtual stereo camera with baseline of 0.5m moves around the house, collecting images of 640×480 pixels. Gaussian white noise with a variance of 1 pixel are added to the points and endpoints of lines in the captured images. Loop detection is disabled to display pose error clearly. $RMSE$ (Root Mean Square Error) of RPE (Relative Pose Error) is the metric to evaluate the performance of our method. Fig. 10 shows an estimated trajectory by our proposed system. The average result of Monte Carlo experiments of 25 runs, is shown in Table I. $RPEtrans1$ and $RPErot1$ are translation and rotation errors obtained in the scene with lots of point features, while $RPEtrans2$ and $RPErot2$ result from the scene containing few points. In the scene with comparable number of points and lines, odometry based on only point feature performs better than one using only lines. The reason may be that re-projection error of an infinite long spatial line is only related to the normal vector of the $Pl\ddot{u}cker$ line coordinates as shown in Section IV-A. So the matched point features produce more constrains than the same number lines. The table shows that the method based on point features has a larger error than on line features in the scene with few points. Our method based on fusion of points and lines outperform than the both.

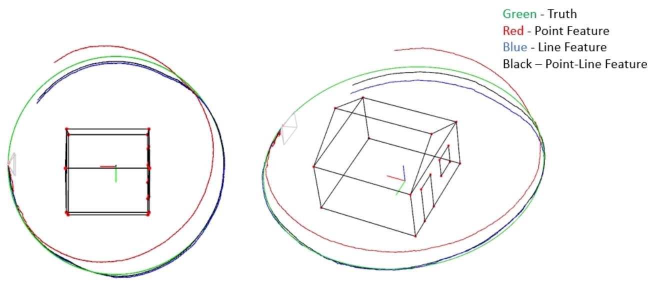


Fig. 10. Top and oblique views of estimated camera trajectory.

TABLE I
RPE of the Methods Based on Different Features

	Point Feature	Line Feature	Point-Line Feature
$RPEtrans1(m)$	0.08702	0.09827	**0.07852**
$RPErot1(rad)$	0.00430	0.00486	**0.00381**
$RPEtrans2(m)$	0.19254	0.09621	**0.08637**
$RPErot2(rad)$	0.00798	0.00481	**0.00408**

2) Real data: The real-world scene experiment is carried on both it3f dataset [6] and KITTI dataset [24]. For a more comprehensive assessment of the approach presented in this article, several open source approaches are compared in this section, including ORB-SLAM2 [1], SLSLAM [6], PLSVO [8] and PL-SLAM presented in this paper. ORB-SLAM2 is a complete point feature based SLAM system that contains map reuse, loop detection and relocation. SLSLAM is based on the straight line feature, constructing scene composed of only straight lines, which is a relatively excellent line based SLAM system. PLSVO is only an odometry using two endpoints to represent the spatial line and performing brute-force matching in the front-end.

Fig. 11 shows images from the it3f dataset, and Fig. 1 shows the results generated from this dataset. Fig. 12 shows the trajectory and map of the camera before and after a loop closure followed by a bundle adjustment. PLSVO has a poor performance on this dataset, so we only compare

Fig. 11. Sample images used in it3f dataset [6].

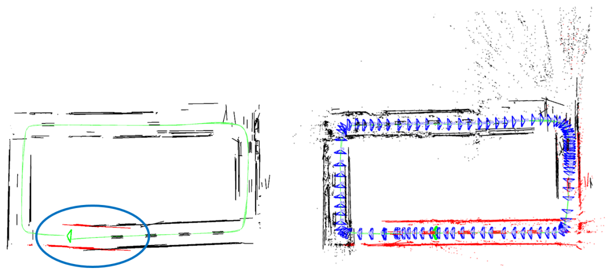


Fig. 12. Results before and after of the loop closure. Left: Results before the loop closure. Right: Results after the loop closure and loop correction.

ORB-SLAM2 and SLSLAM with our proposed PL-SLAM. it3f dataset doesn't provided the ground truth. The degrees of drift before the loop closure are compared. For fair comparison, both ORB-SLAM2 and PL-SLAM disable the loop detection thread, and use the same parameters in point feature extraction. For each image, we extract 1000 point features at 8 scale levels with a scale factor of 1.2.

TABLE II
ERRORS BEFORE LOOP CLOSURE

Method	Errors Before Loop Closure
PL-SLAM	$[-0.3549, 1.4056, -0.0273]^T$
ORB-SLAM2	$[-0.3748, 1.9065, 0.17442]^T$
SLSLAM	$[-0.3141, -0.5455, -0.06449]^T$

Fig. 13 shows the top and side views of the reconstruction results by the three systems without loop closures, respectively. The point with zero coordinates is the starting point and the other is the finishing point. Table II shows the drift before the loop closure (translation in $X(m), Y(m), Z(m)$). It can be observed from the table that PL-SLAM perform better than ORB-SLAM2, which demonstrate the strength of including constraints of straight line. SLSLAM has the best performance, only -0.5455 meters error in the vertical direction. A reason can account for this is that it3f dataset contains low-textured scenarios, reflective white walls, windows and floor etc. At the same time, due to the influence of the ceiling lights, point features prone to be mismatched and bring big errors. In the optimization process of our proposed approach, we don't set different weights to the error terms of points and lines in (12) with consideration of versatility. When the component based on point feature has unstable performance and low accuracy, the proposed system based on combination of point and line features will be affected, which coincides with the synthetic scene experiment.

In terms of time efficiency, the execution time will not increase much because features are extracted in parallel threads. For images with dimensions of 640×480, the feature extraction and stereo matching in ORB-SLAM2 requires

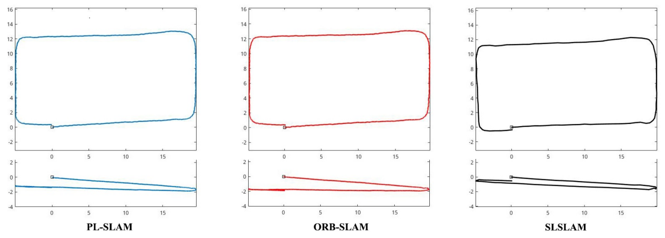


Fig. 13. Comparison results on it3f dataset [6] without loop closure. The top and bottom row show the top and side views of the results.

32.15ms, while our system requires 42.401ms with additional consideration of line features on it3f dataset. Our tracking thread can achieve a performance of 15.1 frame/s, which can satisfy the real-time requirements.

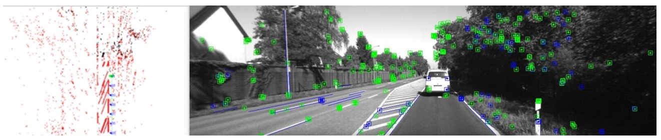


Fig. 14. Results on KITTI dataset. Left: The map composed of point and line features. Right: One frame with extracted point and line features.

We also evaluate our system on KITTI odometry benchmark. Sequences 03, 04, and 10, which have scenarios with lines, are selected. Fig. 14 shows the result on KITTI dataset in our system. In this experiment, we only compare our PL-SLAM with ORB-SLAM2 and PLSVO[1]. Loop detection modules are all disabled for fair comparison. In this experiment, RPE and ATE (Absolute Trajectory Error) are used as evaluation criterion. Table III shows the results of the experiment, where $Trans$ and Rot represent RPE of the translations and rotations respectively. The smallest ATE in each sequence is marked in the table. It is shown that our system has acceptable performance in several sequences. A performance improvement can be achieved compared to the original ORB-SLAM2. PLSVO has a poor performance because of the brute-force matching in data association and accumulated errors.

VI. CONCLUSIONS

To improve the accuracy and robustness of visual SLAM, we present a graph-based approach using point and line features. The spatial line is expressed by the orthonormal representation in the optimization process, which is the compactest and decoupled form. And the Jacobians of re-projection error with respect to the line parameters is also derived out to make a good performance. It is proved that fusing these two types of features will produce more robust estimation in synthetic and real-world scene. Our robust visual SLAM is also able to work in real-time. In the future, we will investigate how to introduce inertial sensors into our system with point and line features.

[1]As the source code of the front-end module in SLSLAM is unavailable, we do not include it in the experiments on KITTI dataset.

TABLE III

RESULTS OF ORB-SLAM, PLSVO AND PL-SLAM ON KITTI DATASET

	PLSVO			ORB-SLAM2			PL-SLAM		
	Trans(m)	Rot(rad)	ATE(m)	Trans(m)	Rot(rad)	ATE(m)	Trans(m)	Rot(rad)	ATE(m)
sequence 03	0.2247	0.0046	13.2415	0.1598	0.0023	2.6638	0.1601	0.0024	**2.6203**
sequence 04	0.2045	0.0019	2.3020	0.1180	0.0015	0.7415	0.1183	0.0017	**0.3663**
sequence 10	0.1809	0.0053	9.0208	0.1143	0.0022	6.3974	0.1166	0.0021	**5.9207**

APPENDIX

This appendix will explain the Jacobian with respect to $\boldsymbol{\delta}_\phi$ in detail. $\boldsymbol{\delta}_\rho$ are set to zeros when computing Jacobian with respect to $\boldsymbol{\delta}_\phi$. With a transformation $\boldsymbol{T}^*$ containing rotation $\boldsymbol{\delta}_\phi$, the new 3D line is denoted as $\boldsymbol{\mathcal{L}}_c^*$:

$$\boldsymbol{T}^* = \exp\left(\boldsymbol{\delta}_\xi{}^\wedge\right)\boldsymbol{T} \approx \left(\boldsymbol{I} + \begin{bmatrix} [\boldsymbol{\delta}_\phi]_\times & \mathbf{0} \\ \mathbf{0}^{\boldsymbol{T}} & 0 \end{bmatrix}\right)\boldsymbol{T} \tag{27}$$

$$\boldsymbol{R}^* = \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{R},\ \boldsymbol{t}^* = \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{t} \tag{28}$$

$$\boldsymbol{\mathcal{H}}_{cw}^* = \begin{bmatrix} \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{R} & \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)[\boldsymbol{t}]_\times\boldsymbol{R} \\ \mathbf{0} & \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{R} \end{bmatrix} \tag{29}$$

$$\boldsymbol{\mathcal{L}}_c^* = \boldsymbol{\mathcal{H}}_{cw}^*\boldsymbol{\mathcal{L}}_w = \begin{bmatrix} \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{Rn} + \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)[\boldsymbol{t}]_\times\boldsymbol{Rv} \\ \left(\boldsymbol{I} + [\boldsymbol{\delta}_\phi]_\times\right)\boldsymbol{Rv} \end{bmatrix}, \tag{30}$$

where $[.]_\times$ denotes the skew-symmetric matrix of a vector. In the process of deducing $\boldsymbol{\mathcal{H}}_{cw}^*$, the properties of rotation matrix $(\boldsymbol{Ra}) \times (\boldsymbol{Rb}) = \boldsymbol{R}(\boldsymbol{a} \times \boldsymbol{b})$, $\boldsymbol{R} \in SO(3)$ is used. Then $\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\phi}$ can be written directly:

$$\frac{\partial \boldsymbol{\mathcal{L}}_c^*}{\partial \boldsymbol{\delta}_\phi} = \begin{bmatrix} \frac{\partial [\boldsymbol{\delta}_\phi]_\times \boldsymbol{Rn}}{\partial \boldsymbol{\delta}_\phi} + \frac{\partial [\boldsymbol{\delta}_\phi]_\times [\boldsymbol{t}]_\times \boldsymbol{Rv}}{\partial \boldsymbol{\delta}_\phi} \\ \frac{\partial [\boldsymbol{\delta}_\phi]_\times \boldsymbol{Rv}}{\partial \boldsymbol{\delta}_\phi} \end{bmatrix} \tag{31}$$

$$= \begin{bmatrix} -\frac{[\boldsymbol{Rn}]_\times \boldsymbol{\delta}_\phi}{\partial \boldsymbol{\delta}_\phi} - \frac{\partial \left[[\boldsymbol{t}]_\times \boldsymbol{Rv}\right]_\times \boldsymbol{\delta}_\phi}{\partial \boldsymbol{\delta}_\phi} \\ -\frac{[\boldsymbol{Rv}]_\times \boldsymbol{\delta}_\phi}{\partial \boldsymbol{\delta}_\phi} \end{bmatrix} \tag{32}$$

$$= \begin{bmatrix} -[\boldsymbol{Rn}]_\times - \left[[\boldsymbol{t}]_\times \boldsymbol{Rv}\right]_\times \\ -[\boldsymbol{Rv}]_\times \end{bmatrix}_{6\times 3} \tag{33}$$

REFERENCES

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[2] C. Rother, "Linear multi-view reconstruction of points, lines, planes and cameras using a reference plane.," in *ICCV*, pp. 1210–1217, 2003.

[3] D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti, "Integration of 3d lines and points in 6dof visual slam by uncertain projective geometry.," in *EMCR*, Citeseer, 2007.

[4] G. Klein and D. Murray, "Improving the agility of keyframe-based slam," in *European Conference on Computer Vision*, pp. 802–815, Springer, 2008.

[5] T. Koletschka, L. Puig, and K. Daniilidis, "Mevo: Multi-environment stereo visual odometry," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4981–4988, IEEE, 2014.

[6] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo slam," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015.

[7] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3934–3942, 2015.

[8] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 2521–2526, IEEE, 2016.

[9] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular slam: Why filter?," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2657–2664, IEEE, 2010.

[10] 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.

[11] 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.

[12] 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, 2010.

[13] 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.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[15] D.-M. Woo, D.-C. Park, S.-S. Han, and S. Beack, "2d line matching using geometric and intensity data," in *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, vol. 3, pp. 99–103, IEEE, 2009.

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

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

[18] Y.-D. Liang and B. A. Barsky, "A new concept and method for line clipping," *ACM Transactions on Graphics (TOG)*, vol. 3, no. 1, pp. 1–22, 1984.

[19] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[20] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based slam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 846–853, IEEE, 2014.

[21] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[23] V. Pradeep and J. Lim, "Egomotion estimation using assorted features," *International journal of computer vision*, vol. 98, no. 2, pp. 202–216, 2012.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.