Line Flow Based SLAM

Qiuyuan Wang, Zike Yan, Junqiu Wang, Fei Xue, Wei Ma, Hongbin Zha

***Abstract*—We propose a method of visual SLAM by predicting and updating line flows that represent sequential 2D projections of 3D line segments. While indirect SLAM methods using points and line segments have achieved excellent results, they still face problems in challenging scenarios such as occlusions, image blur, and repetitive textures. To deal with these problems, we leverage line flows which encode the coherence of 2D and 3D line segments in spatial and temporal domains as the sequence of all the 2D line segments corresponding to a specific 3D line segment. Thanks to the line flow representation, the corresponding 2D line segment in a new frame can be predicted based on 2D and 3D line segment motions. We create, update, merge, and discard line flows on-the-fly. We model our Line Flow-based SLAM (LF-SLAM) using a Bayesian network. We perform short-term optimization in front-end, and long-term optimization in back-end. The constraints introduced in line flows improve the performance of our LF-SLAM. Extensive experimental results demonstrate that our method achieves better performance than state-of-the-art direct and indirect SLAM approaches. Specifically, it obtains good localization and mapping results in challenging scenes with occlusions, image blur, and repetitive textures.**

***Index Terms*—Simultaneous Localization and Mapping (SLAM), Structure from Motion (SfM), Line Segment Extraction and Matching**

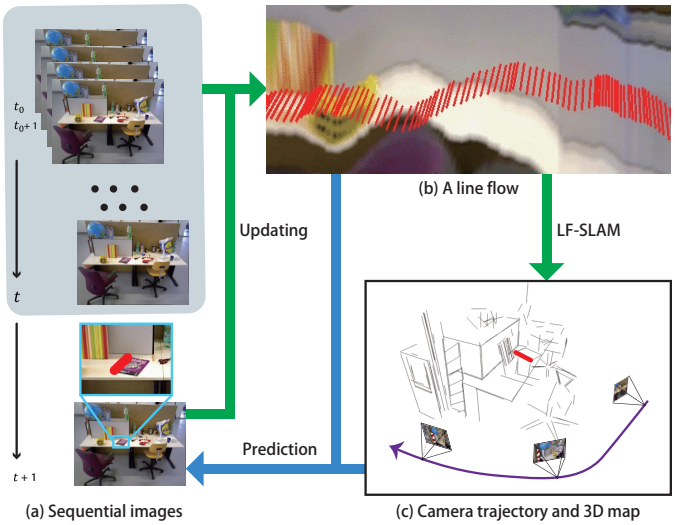


Fig. 1: Overall system structure. We leverage prediction and updating strategy to maintain line flows (b) and SLAM system (c) in sequential images (a).

I. Introduction

SIMULTANEOUS Localization and Mapping (SLAM) aims at continuously estimating camera motion and reconstructing a scene structure in an unknown environment, which is critical for Unmanned Aerial Vehicles (UAVS), autonomous driving, augment reality, and robotics. SLAM systems can be categorized into direct and indirect (feature-based) methods according to their input of minimization [1], [2]. In this paper, we present a Line Flow based-SLAM (LF-SLAM) that effectively exploits the coherence of 2D consecutive observations and 3D maps, as illustrated in Fig. 1.

Direct methods [1]–[3] leverage sensor raw input to optimize photometric errors for pose estimation. Direct methods can cope with untextured scenes and motion blur. Specifically, Direct Sparse Odometry (DSO) [1] samples informative pixels with high intensity gradients and omits the smoothness prior. DSO calibrates all the parameters for the imaging process such as exposure time, lens vignetting, and non-linear response functions. It provides remarkable performance on the datasets under the condition that imaging parameters can be calibrated accurately. However, DSO faces challenges when it is not easy to estimate these parameters. Moreover, it is rather difficult for DSO to perform effective large size bundle adjustment and loop closure because the information at pixel level is not easy to be used for estimating correspondences in consecutive frames.

Indirect methods extract intermediate representation from raw sensor input. Then, camera motion is estimated based on the matching of intermediate representation. Indirect methods can utilize different intermediate representations such as points and lines [4]–[6]. For example, ORB-SLAM [7] considers point features only; and PL-SLAM Stereo [8] and PL-SLAM Mono [9] extract points and lines for pose estimation. Although these methods have achieved great successes on many datasets, spatial-temporal coherence is not considered in these works. We propose LF-SLAM (Fig. 1) to fully model the spatial-temporal coherence of line segments both in 2D and 3D domains. A line flow consists of a set of 2D line segments in consecutive images corresponding to a specific 3D line segment. Line flows are helpful for improving SLAM efficiency and robustness because spatial-temporal coherence can be utilized for feature extraction, feature matching, bundle adjustment, and loop closure.

SLAM systems perform feature extraction when each new image comes in. New features are matched with previous features. Then, optimization is adopted to recover camera poses and 3D features online. To perform real-time processing, many SLAM systems leverage small motion assumption as smoothness priors. For example, ORB-SLAM2 and DSO build grids to decrease candidates in the feature matching step. The coarse poses are obtained by the previous pose. Gaussian-

Qiuyuan Wang, Zike Yan, Fei Xue, and Hongbin Zha are with the Key Laboratory of Machine Perception (Minister of Education), Peking University, Beijing 100871, China (e-mail: {wangqiuyuan, zike.yan, feixue}@pku.edu.cn, zha@cis.pku.edu.cn).

Junqiu Wang is with AVIC Beijing Changcheng Aeronautical Measurement and Control Technology Research Institute, Beijing 100176, China (e-mail: jerywangjq@foxmail.com).

Wei Ma is with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: mawei@bjut.edu.cn).

Newton or Levenberg-Marquardt algorithm is adopted to refine the poses. These strategies do work in practice because temporal smoothness prior is valid in most modern cameras. Furthermore, we found that the estimation results of previous frames are valuable for further processing. The projections of 3D features are predictable in most cases and the previous 2D features can also provide useful information for feature detection in a new input. To be specific, features tend to keep their motion in 2D and 3D domains. The continuity can be guaranteed in most cases even when sudden motions appear. We attempt to utilize the relationships between the different steps in feature-based SLAM system by exploiting line segments with structural cues.

Although significant progress for line segment extraction and matching has been made in recent years [10]–[13], the problems related to line segment extraction and matching are still challenging. Line segment extraction methods usually output line segments with a reliable line direction. However, the endpoints of line segments usually are unstable and inaccurate. Besides, a threshold for line segment extraction is not easily determined for reducing false positives and increasing recalls. An improper threshold breaks a line segment into several small ones. Another possibility is that a line segment is broken by occlusions. Without priors, we can hardly decide which one should be merged. Line segment matching is another thorny problem. Descriptor-based methods have good performance when the endpoints of line segments can be accurately calculated. Unfortunately, images often extract line segments with unstable endpoints. In addition, descriptor-based methods are found not very useful when line segments in nearby image regions have similar appearances, especially in repetitive textures. Structural scenes usually include plenty of repetitive textures. As to the broken line segments, it hard to decide which one should be the correct correspondences. Therefore, line based SLAM systems should solve these problems carefully while exploit the benefit of line segments. For example, broken 2D line segments are calculated separately and lead to redundant 3D line segments in 3D maps. 2D unstable endpoints make the 3D endpoint triangulation results drift.

To deal with these challenging problems for line segment extraction and matching, we consider the coherence of line segment motion in 2D and 3D domains. Since camera motion usually has a smooth trajectory, line segment motion subjects to the constraints of camera motion. A line segment extracted in previous frame provides valuable priors for the extraction in current frame. Several 2D line segments in image sequences corresponding to a 3D line segment can be extracted and related to the 3D line segment. The coherence ensures a highly predictable feature extraction process. Different from other point-line based SLAM systems [8], [9], we predict and update line flows according to the spatial-temporal coherence in consecutive frames.

To be specific, independent 2D line detection at each frame is unreliable due to noises, occlusions, and visual ambiguities (such as the edge of a cylinder-like object). Triangulation based on these observations might lead to incomplete or erroneous 3D line segments. We carefully maintain the spatial-temporal consistency, and construct a structural and informative map consisting of 3D point and line segments. The confidence of each line segment is evaluated according to its stability in the sequence, and the localization results of previous frames are exploited to support the discovery of these correspondences.

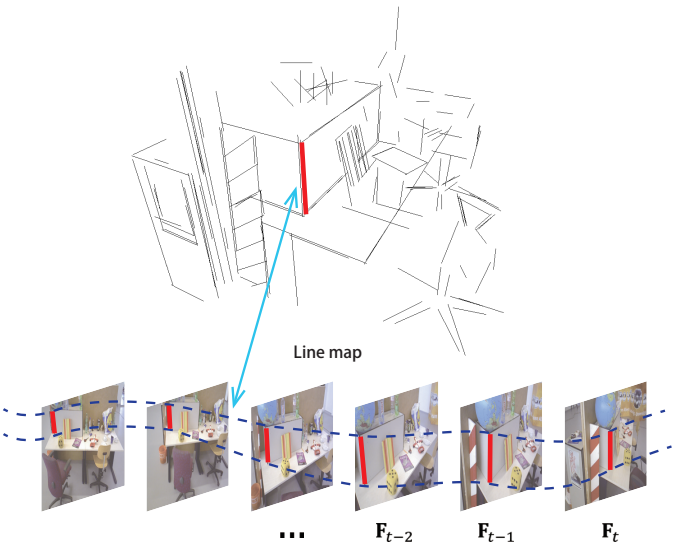


Fig. 2: A line flow in sequential images. All the line segments in the line flow correspond to the same 3D line.

We find corresponding 2D and 3D line segments based on predicting and updating line flows in image sequences, as illustrated in Fig. 2. The line flows are embedded into a Bayesian network for efficient and robust SLAM. The contributions of this work are twofold:

- We propose line flows to fully model the spatial-temporal coherence of line segments in 2D and 3D domains. We create, insert, merge, and discard line segments at each frame based on line flow representation, which bridges the different stages of a SLAM system, including feature extraction, feature matching, triangulation, and optimization. The structure consistency can be stably preserved in a predicting-updating fashion;
- We propose a novel monocular SLAM method based on line flow representation. The coherence in line flows leads to accurate and efficient camera localization and mapping in challenging scenarios such as occlusions, image blur, and repetitive textures. In addition, our system can generate a precise and visually appealing 3D maps on-the-fly.

We organize this work as follows. We introduce related work in Section II. Line flow representation and the SLAM system pipeline are described in Section III. Then, the details of line flow tracking are given in Section IV and line flow mapping in Section V. Extensive experiments including comparison with state-of-the-art SLAM approaches are given in Section VI. Finally, we conclude this work in Section VII.

II. Related Work

Line flow-based SLAM has relationships with line segment detection and matching, 3D line reconstruction, and visual

SLAM. There is a large literature on each kind of the tasks, we briefly review the most relevant works as follows.

A. Line Segment Detection and Matching

Detecting and matching line segments from images is a classical problem, which is essential for diversified 3D vision tasks, e.g., line reconstruction and SLAM. Hough transform [12], [14] is widely used for line detection by voting to select valid lines through edge pixels. These methods utilize the positions of edge pixels to locate potential lines. However, it is computationally expensive, hence, not suitable for a real-time applications. Hough transform is not suitable for a SLAM system due to the high computational complexity. An efficient alternative is to locate line segments by aggregating adjacent pixels with similar gradient orientations [10], [15], [16]. Unfortunately, the gradient pseudo-ordering and the region growing implementation fail to take previous information into consideration, which increases unnecessary computational overhead. [13] gives a comprehensive analysis of line segments and presents a novel linelet representation for line segment detection. However, the high computational cost restricts its practical usage. Recently, learning based detection methods [17], [18] have emerged and outperform most of existing methods. They also have high computational cost and rely on GPUs. In this work, we consider the spatial-temporal coherence of line segments by integrating LSD [10].

Different methods have been proposed for line segment matching. Most of the existing methods are designed for two-view matching. MSLD [19] calculates the mean and standard deviation statistics of each line for matching. The approaches in [20], [21] match lines according to the neighboring feature descriptors. LBD [11] builds a relational graph and leverages local appearance and geometry relationship for line matching. The algorithm shows remarkable results and is widely adopted in reconstruction/SLAM [8], [9], [22]. In our work, spatial-temporal coherence is exploited by line flows to guide efficient and effective line segment matching.

B. Visual SLAM

LineSLAM [23] takes lines as the basic feature and performs unscented Kalman filter (UKF) as a tracking algorithm. Their experiments show good performance in simulation settings. StructSLAM [6] introduces the conception of line that encodes global orientation for vanishing points calculation. This method works well in Manhattan-like environments. Li *et al.* [5] also leverage structural line features to obtain accurate camera poses. Solà *et al.* [24] conduct a comprehensive study to better understand the impact of different point and line parametrization on EKF-based SLAM. However, the complexity of the EKF algorithm depends on the square number of the landmarks, which can lead to unacceptable computational cost for real-time applications. Besides the aforementioned methods with single cameras, a few works leverage stereo cameras [8], [25], [26] or RGB-D cameras [27] for better geometric cues. Zhao *et al.* [22] present a good line cutting approach.

Dong *et al.* [28] propose a monocular SLAM method using point and line features for graph optimization. It gives higher accuracy than the EKF-based SLAM method. PL-SLAM Mono [9] proposes a monocular SLAM system based on point and line features and achieves better performance than many other methods. PL-SLAM Mono extracts line segments and their descriptor for matching. Therefore, the complexity of line segment detection and matching is relatively high since all the pixels in an image are visited and LBD descriptors are computed in each image. As to bundle adjustment, 3D lines from local map project onto images to search for correspondences. Bundle adjustment utilizes points and lines together in point-line re-projection errors. Loop closure only uses points in PL-SLAM Mono [9], since matching lines across the whole map is too computationally expensive. Our method achieves highly efficient SLAM with high accuracy and concise online 3D mapping thanks to the unified framework by fully using spatial-temporal cues induced by our line flow representation. The sufficient number of line matching results strengthens the constraints for bundle adjustment. After we detect loop closure, line flows are added to optimize parameters based on reliable and long-term 2D line segments recorded in line flows.

C. 3D Line Reconstruction

3D line reconstruction has been a research topic for decades. An early attempt [29] uses a graph-based description of line segments for 3D line reconstruction given two stereo images. [30] is the first method to introduce a line-based SfM pipeline. However, the strong constraint within the trifocal tensor restricts the reconstructed lines. [31] imposes global topological constraint using neighboring connections between line segments for outlier rejection, but it requires camera pose ground truth. Recently, [32] leverages the collinear property from extracted 2D line segment cues to fit 3D line. The state-of-the art approach [33] formulates line reconstruction procedure as a graph-clustering problem and achieves promising results. Unfortunately, it assumes a known camera pose which is not suitable for SLAM.

All the aforementioned methods run offline, requiring the entire image sequence in a batch mode. A few incremental 3D

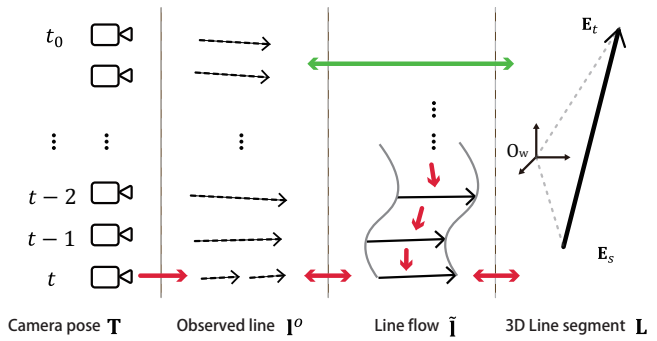


Fig. 3: Coherence within a line flow. Traditional methods only establish correspondences between observed line segments $\mathbf{l}^o$ and 3D lines $\mathbf{L}$ at each view (green arrows). In contrast, the line flow $\tilde{\mathbf{l}}$ establishes additional temporal correlation between line segments at different views that correspond to the same 3D line segment (red arrows). As a result, line parameters are constrained with priors. Broken line segments can be merged based on the priors.

line reconstruction methods are also proposed. [34] complements the Plücker coordinate with the Cayley representation, but fails to handle the non-trivial line matching and translatory motion drift issues. [20] tries to deal with unstable endpoints, and gains efficiency by decoupling translation and rotation. However, it relies on a strong assumption that two parallel lines are orthogonal to a third one, which restricts its practical usage.

III. Line Flow Representation & SLAM pipeline

We will introduce line flow in section III-A and the SLAM pipeline in section III-B. We model line flows on monocular input. Although stereo systems can track, segment, or reconstruct objects by using depth information obtained based on the input, monocular cameras have a few advantages over stereo systems: i). they do not need extrinsic calibration between two cameras; ii). they are cheaper; iii). they usually have a better field of view because no view intersection is needed.

A. Line Flow Representation

As illustrated in Fig. 2, a line flow $\tilde{\mathbf{l}}$ is defined as the sequence of 2D line segments corresponding to a 3D line $\mathbf{L}$, and is maintained from the first observation at time t_0 to current frame:

$$\tilde{\mathbf{l}} = \{\mathbf{l}_{t_0}, \mathbf{l}_{t_0+1}, \cdots, \mathbf{l}_t\}. \tag{1}$$

We parameterize a 2D line segment $\mathbf{l}$ by (θ, l, x, y), including line orientation θ, length l, and middle point position (x, y). We adopt the orientation representation in [10]. Following [25], [33], an oriented 3D line segment $\mathbf{L}$ is parameterized using a 3D infinite line representation $(\mathbf{n}^T, \mathbf{v}^T)^T$ [35] in the Plücker coordinates with initial and terminal endpoints $\mathbf{E}_s, \mathbf{E}_t$. Note that i). both 2D line segments and 3D line segments have directions; ii). 3D endpoints are always on 3D lines.

Following the definition, we summarize 3 properties:

1) **The relationship between a 2D line segment and its corresponding 3D line segment.** Each 2D line segment $\mathbf{l}_{t_i}$ in the line flow is collinear with the 2D projection of 3D line segment. Note that the 2D projection of a 3D line segment can be different from the 2D line segments in incoming images due to occlusions. Based on this property, we design the back-end optimization module of our SLAM system (in Section III-B & Section V).
2) **The relationships of 2D line segments in a line flow.** 2D Line segments observed in consecutive frames are constrained by image motion. When a camera moves, the spatial-temporal coherence provides constraints for the line segments. The constraints are effective in consecutive frames when the corresponding 3D line segment can be observed. These constraints provide helpful information for line flow extraction. We design the predicting module for leveraging the constraints (in Section IV-A).
3) **Line segment properties.** A line flow inherits the properties of line segments: a). reliable line segment directions; b). each line segment is observed as a set of pixels with similar gradient orientations; c). the gradient orientations are perpendicular to the line segment. Based on these properties, we can merge certain sub-segments when they have similar line directions (in Eq. (7)). We can guide line extraction by grouping similar gradient orientations (in Section IV-B).

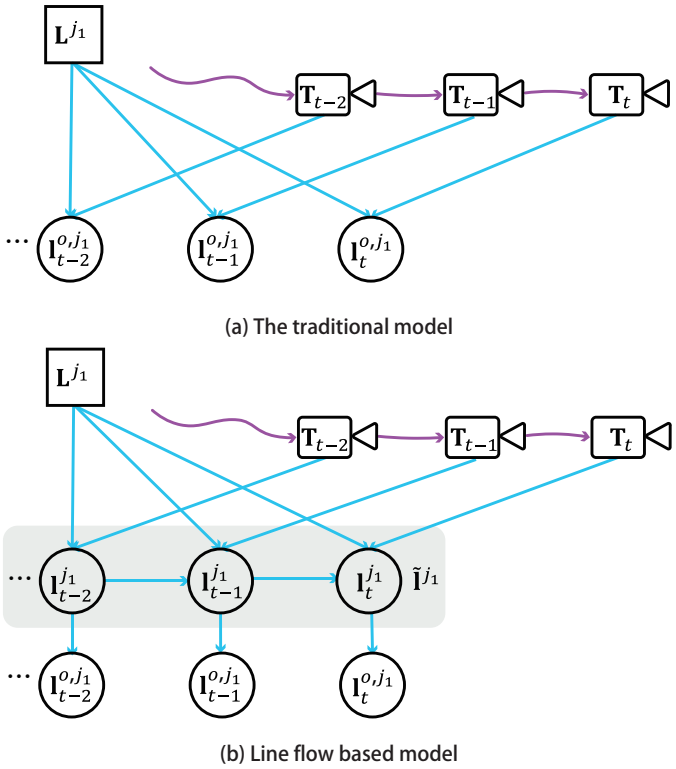


Fig. 4: (a) Traditional line based SLAM system, (b) Line flow based SLAM. Compared to the traditional model (a), our line flow based model (b) establishes temporal correlations for each line segment and camera poses.

We show how to build line flows using coherence in Fig. 3. Line segment extraction in single frames can give incorrect results due to occlusions, image blur, and noises. We consider the coherence of line segments in consecutive frames to partially solve this problem.

Finding reliable line segment correspondences is a prerequisite for successful triangulation. Traditional line-based SLAM methods only consider the correspondences of line segments in two frames based on line segment detection and matching results [8], [9], [25]. Unfortunately, finding reliable correspondences using detection and matching is very challenging because partial occlusions split a line into several segments; image noises lead to unstable line endpoints; visual artifacts bring about false-positive line segments; a scene with repetitive textures may cause severe matching ambiguities. Our line flow exploits spatial-temporal consistency to establish correspondences between sequential line segments for 2D line prediction, extraction, updating and matching.

Our approach has several advantages:

1) We do not need an explicit line segment descriptor because we predict and update line segments in a line flow using the coherence;
2) The constraints in a line flow are utilized for reliable and efficient 2D line segment extraction. Extracting line

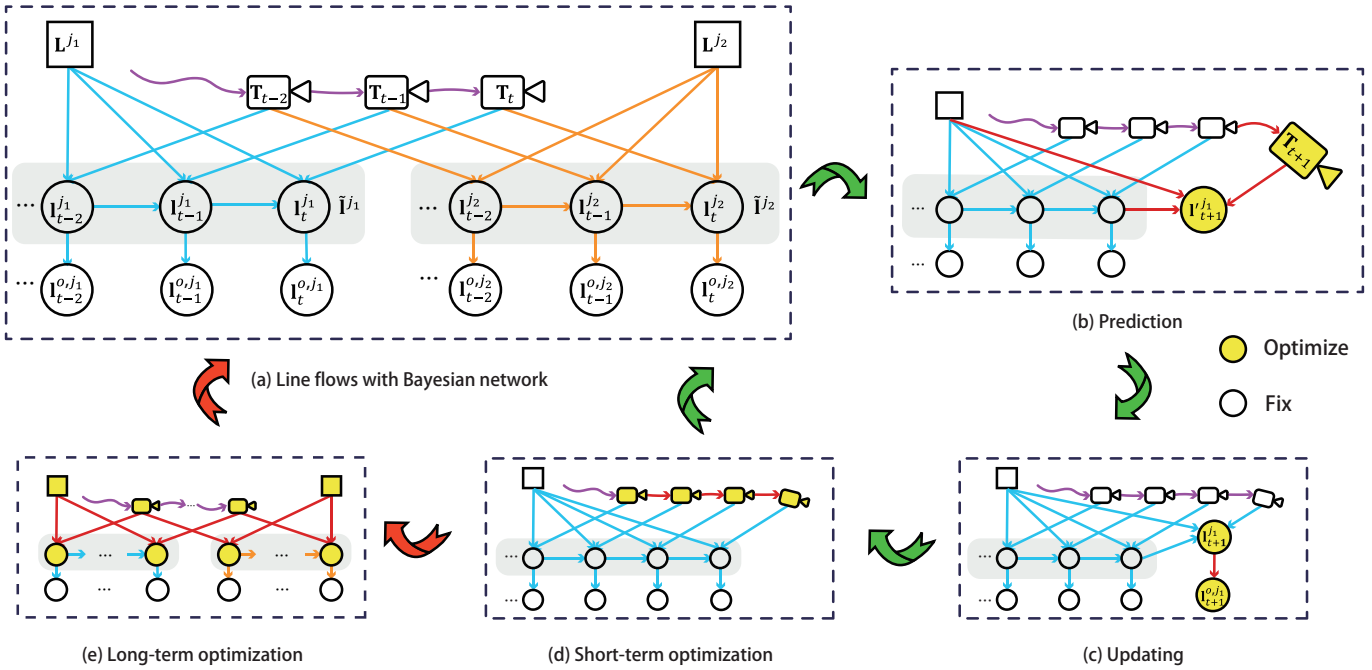


Fig. 5: We model the problem using a Bayesian network (a), which consists of line segment prediction (b), line flow updating (c), short term optimization (d), and long term optimization (e). The yellow nodes denote variables to be updated, and white nodes are fixed. The dark green cycle is processed frame-by-frame, while the red cycle is performed only on key frames.

segments and finding their correspondences in this way are robust against occlusions and outliers;

3) The reconstructed line map rejects false positive observations. For example, the sides of a cylinder do not formulate reasonable triangulations in consecutive frames. Therefore, we discard such lines for generating reasonable maps.

B. Line Flow based SLAM

As illustrated in Fig. 4, we construct a graph model for line flows. Each line flow $\tilde{\mathbf{l}}^j$ corresponds to a 3D line $\mathbf{L}^j$, and the correlation is constrained by camera poses $\mathbf{T} \in \mathrm{SE}(3)$ and the observations $\mathbf{l}_t^o$. At the very beginning of a SLAM, we take the camera coordinate of the first frame as the world coordinate. The relative transform from the world to t-th frame is denoted by $\mathbf{T}_t \in \mathrm{SE}(3)$.

As illustrated in Fig. 5 (a), the probabilistic relationships between the variables and the observations can be modeled by a Bayesian network. The joint probability can be described as:

$$\begin{aligned} \mathrm{P}(S) \propto \mathrm{P}(\mathbf{T}_0) \prod_{t,j,k} & \mathrm{P}(\mathbf{l}_t^{o,j} | \mathbf{T}_t, \mathbf{L}^j, \mathbf{l}_{t-1}^j) \\ & \mathrm{P}(\mathbf{p}_t^{o,k} | \mathbf{T}_t, \mathbf{P}^k) \mathrm{P}(\mathbf{T}_t | \mathbf{T}_{0\ldots t-1}), \end{aligned} \tag{2}$$

where S denotes the entire set of all unknown variables including camera poses $\{\mathbf{T}\}$, 3D line segments $\{\mathbf{L}\}$, 2D line flows $\{\tilde{\mathbf{l}}\}$, and 2D line observations $\{\mathbf{l}^o\}$; $\mathrm{P}(\mathbf{T}_0)$ is the camera pose prior; $\mathrm{P}(\mathbf{l}_t^{o,j} | \mathbf{T}_t, \mathbf{L}^j, \mathbf{l}_{t-1}^j)$ is the line measurement model; $\mathrm{P}(\mathbf{p}_t^{o,k} | \mathbf{T}_t, \mathbf{P}^k)$ is the point measurement model; $\mathrm{P}(\mathbf{T}_t | \mathbf{T}_{0\ldots t-1})$ is the camera motion model. The probabilistic formulation Eq. (2) can be transformed to an energy function and can be solved efficiently with a non-linear optimization method [25], [36], [37].

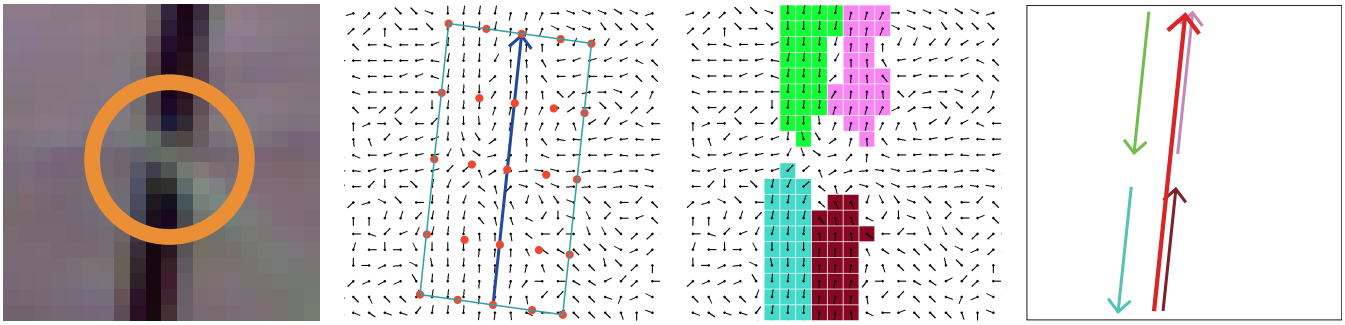
The proposed pipeline is illustrated in Fig. 5 in an incremental fashion. We design a front-end (green arrows) running incrementally at frame-rate for optimizing line flows and camera poses; and a back-end (red arrows) running only for key-frames to optimize 3D line segment mapping and camera poses. Fig. 5 (a) is the graph state at time t. We predict the 2D projection of the 3D line segment corresponding to a line flow in a new frame. Then, we update the line flow according to the constraints and the information in a new observation. We optimize the camera pose $\mathbf{T}_{t+1}$ by fixing 2D features and 3D landmarks in short-term optimization. We also perform long-term optimization for 3D line segments in the map in the back-end.

IV. Line Flow Tracking

In this section, we introduce line flow prediction and updating. Both of them are performed in the front-end. The main issue of the tracking module is how to maintain line flows when new frames are captured. First, we predict each line segment by finding the correspondences of line segments in 2D and 3D domains based on 2D and 3D motions. A line flow is then updated based on a new observation. Unlike other line-based SLAM systems, we can find reliable correspondence using spatial-temporal constraints and line geometric properties. This approach achieves better performance than descriptor-based matching.

A. Prediction

We infer a new line segment based on motion coherence. The first derivatives of both 6-DoF camera motion differences



(a) RGB image (b) Level-line field (c) Line support regions (d) Line fusion

Fig. 6: Guided line extraction process during occlusions (a). A few seeds (red) are uniformly sampled (b) within the rectangular area around the predicted line segment (dark blue), where level-line field [38] is illustrated as small black vectors. Four line support regions are found using the seeds (c), and four line segment candidates are subsequently found. Line segment that is the most collinear to the predicted line is chosen as the best candidate. Then, we fuse it with other collinear candidates. In this way, we can extract the complete line segment even when occlusion occurs (d), and we can distinguish line segments on both sides according to our similarity criteria. The line segment on the left side will be extracted with the corresponding predicted line.

$\mathbf{m_T}$ and 2D line parameters differences $\mathbf{m_l}$ are assumed to be constant within a sliding window because large derivatives are usually introduced by noises. Starting from t_w-th frame, we have

$$\begin{aligned} \mathbf{T}_{t-1} &= \mathbf{T}_{t-2}\mathbf{m_T}, \\ \mathbf{T}_{t-2} &= \mathbf{T}_{t-3}\mathbf{m_T}, \\ &\cdots \\ \mathbf{T}_{t_w+1} &= \mathbf{T}_{t_w}\mathbf{m_T}. \end{aligned} \tag{3}$$

Similar to pose motion, we have

$$\begin{aligned} \mathbf{l}^j_{t-1} &= \mathbf{l}^j_{t-2} + \mathbf{m}^j_{\mathbf{l}}, \\ \mathbf{l}^j_{t-2} &= \mathbf{l}^j_{t-3} + \mathbf{m}^j_{\mathbf{l}}, \\ &\cdots \\ \mathbf{l}^j_{t_w+1} &= \mathbf{l}^j_{t_w} + \mathbf{m}^j_{\mathbf{l}}, \end{aligned} \tag{4}$$

We calculate these two motions by solving a least-square problem [37].

Line segment prediction based on 2D coherence. 2D prediction is achieved by modeling the 2D motion $\mathbf{m}^j_t$ of each line flow $\tilde{\mathbf{l}}^j$, which is defined as the parameter changes of the corresponding line segments $\mathbf{l}^j_{t-1}$ and $\mathbf{l}^j_t$ in two consecutive frames. With the aforementioned constant velocity assumption, the 2D motion model of each line flow can be fitted with previous estimation results, and the predicted line segment $\mathbf{g}^j_t$ using 2D coherence is defined as:

$$\mathbf{g}^j_t = \mathbf{l}^j_{t-1} + \mathbf{m}^j_{\mathbf{l}}. \tag{5}$$

Line segment prediction based on 3D coherence. 3D prediction is achieved by projecting two endpoints of each 3D line segment $(\mathbf{E}^j_s, \mathbf{E}^j_t)$ into current frame with a predicted camera pose $\mathbf{T}_t = [\mathbf{R}_t|\mathbf{t}_t]$, where the pose prediction is calculated in a similar way as the 2D motion model. Then, the predicted line segment $\mathbf{h}^j_t$ using 3D coherence is calculated by:

$$\begin{aligned} \mathbf{e}^j_s &= \lfloor \mathbf{K}_p(\mathbf{R}_t\mathbf{E}^j_s + \mathbf{t}_t) \rfloor, \\ \mathbf{e}^j_t &= \lfloor \mathbf{K}_p(\mathbf{R}_t\mathbf{E}^j_t + \mathbf{t}_t) \rfloor, \\ \mathbf{h}^j_t &\leftarrow (\mathbf{e}^j_s, \mathbf{e}^j_t), \end{aligned} \tag{6}$$

where $\mathbf{K}_p$ is intrinsic matrix for point features, $\lfloor \cdot \rfloor$ represents the transformation from the homogeneous coordinate to the 2D coordinate, and $\mathbf{e}^j_s, \mathbf{e}^j_t$ are two projected endpoints.

Line segment prediction based on 2D and 3D coherence. Line segment prediction using 2D coherence is performed for all the frames. This prediction is useful when 3D line segments are not stable (e.g., during the initialization of a line flow). However, 2D motion can also be unreliable due to large displacement. Fortunately, line segment prediction using 3D coherency is helpful for improving the accuracy since camera trajectories are usually smooth. In general, a long line segment potentially indicates accurate line directions. To integrate these two kinds of prediction results, we fuse them using a length weighting operation:

$$\mathbf{l}'^j_t = \frac{l_{\mathbf{h}}}{l_{\mathbf{h}} + l_{\mathbf{g}}}\mathbf{h}^j_t + \frac{l_{\mathbf{g}}}{l_{\mathbf{h}} + l_{\mathbf{g}}}\mathbf{g}^j_t, \tag{7}$$

where $l_{\mathbf{h}}$ and $l_{\mathbf{g}}$ are the length of $\mathbf{h}^j_t$ and $\mathbf{g}^j_t$. A long line segment has more influences for line fusion results, which means that the fused line angle leans towards the longer one.

The purpose of line segment prediction is to accelerate extraction and matching processes and to refine line segments, e.g., fusing some broken line segments. In addition, when we cannot observe line segments due to image blur or false positive detection, we use the predicted line segments to keep the continuity for the corresponding line flow. These line segments can be integrated into the corresponding line flow when new complete observations are available. We need to refine line flows since predicted line segments can be inaccurate.

B. Updating

We update a line flow by considering the line segments in a line flow, line segment prediction and new observations. The updating has 3 steps: guided line extraction, line flow creation, and line flow management.

Guided line extraction. The extraction process is illustrated in Fig. 6. We highlight the major difference between our guided line extraction method and other methods [9], [22] that extract line segment independently on each frame using LSD [10].

We give a brief description of LSD [10]. First, the gradient of each pixel is calculated. The gradients in a region form a level-line field [10]. All the pixels are sorted according to gradient magnitudes. The pixels with high gradient magnitudes tend to form line segments. Starting from the highest gradient pixel as the seed, LSD performs region growing according to the expansion criterion to form line support regions. The expansion criterion is that a pixel must not be visited before and the orientation difference between the angles of the pixels and the angles of the region should be lower than a threshold. The rectangular approximation is applied to form a line segment. Finally, NFA (the Number of False Alarms) is calculated and refinement is performed for better line segment.

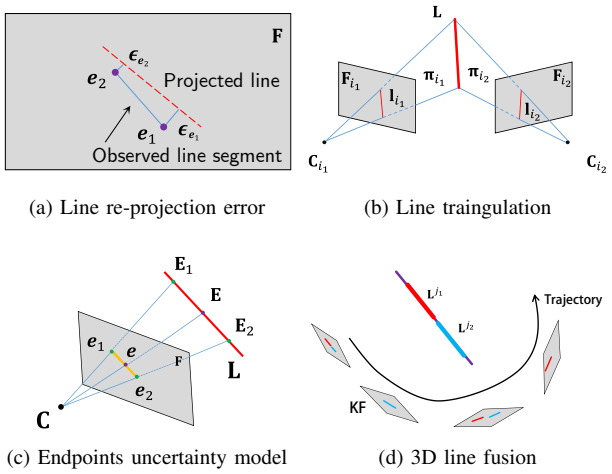
We extract line flows considering the coherence in spatial-temporal domain by adapting the LSD method. With line segment prediction, we do not have to detect line segments in a new frame from scratch. The nearby regions of predicted line segments are taken into consideration. This strategy makes our methods efficient because these edge pixels are just a few percent of the images.

To form a line support region, we do not need to sort gradient magnitudes. We adopt a rectangular search region expanded from the predicted line segment. Seeds of line segments are sparsely sampled near the predicted line segments. For each seed, we perform region growing to find the corresponding line segment regions. Then, we extract line segment candidates from each line support region by approximating rectangles according to gradient orientations. Several line segment candidates are calculated based on the rectangles. We select the one that has the best collinearity with the predicted line segment as the best candidate. Other candidates can be fused with the best candidate when their angular difference is less than a threshold based on the collinearity of these two line segments by performing the fusion operation in Eq. (7). With the guidance provided by the prediction, our line extraction is more reliable by rejecting false-positive lines Moreover, it can fuse line segments broken by partial occlusions.

Line flow creation. A line flow is created when we bootstrap the system or when a new line segment is observed. We run the LSD algorithm [10] for line extraction every N_{cl} frames over remaining unvisited pixels. An experiment (in Section VI) is performed to study the effect of N_{cl}. We create a line flow when a new observed line segment comes in. Since no prior motion is available for the corresponding line segment, we directly use the first line segment as a predicted line segment. We perform a KLT [39] for each seed generated by the predicted line segment. Then, the corresponding seeds are adopted to guide line extraction.

Line flow management. We need to carefully maintain stable coherences and enhance robustness. When the 2D line segments corresponding to a line flow are temporarily unobservable in a few frame, we set a reserving period α before terminating this line flow. Within the reserving period, the line flow is still kept alive with predicted line segments as the observation.

One 3D line might be split into multiple line flows due to occlusions. We check all line flows and fuse the line flows when more than half of their line segments are collinear and their recent line segments are overlapping with certain pixels (set to 5 pixels). If these two line flows are correspond to the same 3D line, we merge the line segments using the merging operation (Eq. 7). To accelerate this step, we utilize the collinearity of two line segments in current frame. The angle-grid based method is adopted. We set 30° as the grid size and put line segments into the corresponding grids between 0° to 360°. Note that the number of line flow is not very large, therefore line flow merging has little negative effect on the real-time performance.



(a) Line re-projection error (b) Line traingulation

(c) Endpoints uncertainty model (d) 3D line fusion

Fig. 7: 3D Line operations in the line flow include triangulating and updating a 3D line.

Although the orientation of a line segment is relatively stable, the lengths can vary significantly across different viewpoints. We refine the length considering the historical lengths:

$$l_t^j = \max(\frac{1}{\beta}\bar{l}_t^j, \min(\beta\bar{l}, l_t^{o,j})), \tag{8}$$

where $l_t^{o,j}$ denotes the length of an observed line segment; $\bar{l}_j^t$ denotes the mean length of the latest line segments stored in the corresponding line flow; β controls the rate of line length change.

V. Line flow based SLAM System

SLAM is carried out using line flows with spatial-temporal coherence. The temporal coherence can be utilized to further constrain localization and mapping. Camera pose is estimated for each frame using the short-term optimization, while the 3D map and poses are jointly refined at each key-frame through the long-term optimization in a parallel back-end. 3D line segments as an implicit part of line flows provide geometric constraints for filtering outliers that are found in 2D but without real correspondences in 3D, e.g., the edge of cylinder objects. Moreover, we can search more correspondences among line flows according to their 3D line segments.

A. Short-term Optimization

We can easily find 2D line segments and the corresponding 3D line segment through line flows $\tilde{\mathbf{l}}$. Thus, by adding 2D-3D

point constraints $\mathbf{p}$, we jointly solve camera poses within a sliding window by minimizing an energy function:

$$C_s = \min_{\{\mathbf{T}\}} \sum_{\{\tilde{\mathbf{l}}\},\{\mathbf{p}\},\{\mathbf{T}\}} \rho(\boldsymbol{\epsilon}_\mathbf{l}^T \boldsymbol{\Sigma}_\mathbf{l} \boldsymbol{\epsilon}_\mathbf{l} + \boldsymbol{\epsilon}_\mathbf{p}^T \boldsymbol{\Sigma}_\mathbf{p} \boldsymbol{\epsilon}_\mathbf{p} + \boldsymbol{\epsilon}_\mathbf{T}^T \boldsymbol{\Sigma}_\mathbf{T} \boldsymbol{\epsilon}_\mathbf{T}), \tag{9}$$

where $\boldsymbol{\epsilon}_\mathbf{l}$ and $\boldsymbol{\epsilon}_\mathbf{p}$ are the re-projection errors in terms of lines and points respectively; $\boldsymbol{\epsilon}_\mathbf{T}$ the smoothness error of the camera trajectories. $\boldsymbol{\Sigma}_\mathbf{p}$, $\boldsymbol{\Sigma}_\mathbf{l}$, $\boldsymbol{\Sigma}_\mathbf{T}$ are the covariance matrices for points, line segments and relative poses since we don't consider the relationship between these parameters. We set these covariance matrices to identity matrices. A Huber function $\rho(\cdot)$ is adopted to increase robustness.

The reprojection error is modeled following [25]. We define the line re-projection error $\boldsymbol{\epsilon}_\mathrm{l}$ as the distance from two observed endpoints $\mathbf{e}_1, \mathbf{e}_2$ to the projected line segment of 3D line $\mathbf{L}$, which is illustrated in Fig. 7 (a). The error functions of lines and points are defined as:

$$\boldsymbol{\epsilon}_\mathbf{l} = \frac{1}{\sqrt{l_x^2 + l_y^2}} [\mathbf{e}_1 | \mathbf{e}_2]^T \mathbf{K}_l \left[\mathbf{R} | [\mathbf{t}]_\times \mathbf{R}\right] \mathbf{L}, \tag{10}$$

$$\boldsymbol{\epsilon}_\mathbf{p} = \mathbf{p} - \lfloor \mathbf{K}_p (\mathbf{R}\mathbf{P} + \mathbf{t}) \rfloor, \tag{11}$$

$$\mathbf{K}_l = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ -f_y x_0 & -f_x y_0 & f_x f_y \end{bmatrix}, \mathbf{K}_p = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

where $\mathbf{K}_l$ is the intrinsic matrix for lines [25], $\mathbf{K}_p$ is the intrinsic matrix for points. f_x and f_y are the focal lengths and $(x_0, y_0)^T$ is the principle point; $[\mathbf{t}]_\times$ is the antisymmetric form of $\mathbf{t}$; $\mathbf{p}$ and $\mathbf{P}$ denote 2D and 3D points, respectively; Assuming that the projection of 3D line segment $\mathbf{L}$ on 2D image is (l_x, l_y, l_z), $\frac{1}{\sqrt{l_x^2 + l_y^2}}$ is a term for distance normalization; $\lfloor \cdot \rfloor$ represents the transformation from the homogeneous coordinate to the 2D coordinate.

Trajectory smoothness term serves as a regularization to ensure a smooth camera trajectory. We assume constant rigid motion between successive poses to enforce temporal consistency:

$$\boldsymbol{\epsilon}_\mathbf{T} = \log_{\mathrm{SE}(3)} \left(\mathbf{T}_{t-1} \mathbf{T}_{t-2}^{-1} \mathbf{T}_{t-1} \mathbf{T}_t^{-1}\right). \tag{13}$$

B. Long-term Optimization

The long-term optimization is performed at each keyframe to jointly refine line flows, 3D map, and camera poses. We follow [7] to manage keyframes. The management of 3D line map includes 3D line creation, outlier rejection, merging and updating based on line flow representation. We triangulate the latest two line segments within each line flow, then, reject outliers and merge correlated 3D lines to maintain a reliable line map. After that, we make a local bundle adjustment. An optional global optimization with loop detection [40] is performed.

Line flow triangulation. Triangulation is performed when a line flow has survived at least two key frames and the angle between two planes π_i and π_j is greater than 1° (Fig. 7 (b)). A 3D line $\mathbf{L} = (\mathbf{n}_L^T, \mathbf{v}_L^T)^T$ is represented by intersecting two planes generated from corresponding 2D line segments and the camera centers:

$$\begin{aligned} \mathbf{L} &\leftarrow \begin{cases} \mathbf{n}_L = d_{i_2} \mathbf{n}_{i_1} - d_{i_1} \mathbf{n}_{i_2}, \\ \mathbf{v}_L = \mathbf{n}_{i_2} \times \mathbf{n}_{i_1}, \end{cases} \\ \boldsymbol{\pi}_i &\leftarrow (\mathbf{n}_i, d_i) = \left[\mathbf{R} \mid \mathbf{t}\right]^T \mathbf{K}_l^T \mathbf{l}. \end{aligned} \tag{14}$$

Then, we check the projection errors between 3D line segment $\mathbf{L}$ and 2D line segment within the corresponding line flow. When the number of the outliers is greater than 90% of the number of 2D line segments, we remove the line flow.

We maintain 3D line segment endpoints after a successful triangulation. We compute the mean of back-projected 3D endpoint for each 2D line segment as the initial 3D line endpoints. We leverage 2D start/terminal endpoints to calculate 3D start/terminal endpoints. Then, when each new line segment comes in, we update 3D start/terminal endpoints by an averaging operation. The averaging operation is simple and highly efficient because the outlier rejection operation can remove unstable endpoints, and the line flow merging process fuses broken line segments.

Outlier rejection. As illustrated in Fig. 7 (c), if a 3D line segment is far from the camera, a small disturbance on the image plane will lead to a large deviation, which makes the 3D line segment highly unstable. To handle this issue, we calculate the uncertainty $\mathrm{U}_\mathbf{e}$ according to the distance between 2D and 3D line segment endpoints:

$$\mathrm{U}_\mathbf{e} = \frac{\|\mathbf{E}_1 - \mathbf{E}_2\|}{\|\mathbf{e}_1 - \mathbf{e}_2\|}, \tag{15}$$

where the 2D pixels $\mathbf{e}_1, \mathbf{e}_2$ are 0.5 pixel away from $\mathbf{e}$ along the line direction, and $\mathbf{E}_1, \mathbf{E}_2$ are the projected 3D points accordingly.

Larger uncertainty implies more unstable endpoints. In practice, we update the mean μ_U and standard deviation σ_U of all line segment endpoints every 10 key-frames, and reject a 2D line segment if the uncertainty of at least one endpoint exceeds $\mu_\mathrm{U} + 3\sigma_\mathrm{U}$. We further find that erroneous line matching tends to generate long 3D line segments. Meanwhile, a 3D line on a plane parallel to the optical axis has unstable endpoints, and is usually observed as a short line segment. Hence, our uncertainty criteria can reject these two kinds of 3D lines to guarantee high reliability.

Line flow merging. As illustrated in Fig. 7 (d), a 3D line might be visible again after the ending of the corresponding line flow due to certain reasons such as occlusion, out of sight, severe blurring. These two 3D lines tied to two separate line flows should be merged. We merge two 3D lines $\mathbf{L}^{j_1}$ and $\mathbf{L}^{j_2}$ with their line flows if they are collinear and overlapping. Assuming that the number of 2D observed line segment set of $\mathbf{L}^{j_1}$ is larger than the number of 2D observed line segment set of $\mathbf{L}^{j_2}$, we merge the set of 2D observed line segment of $\mathbf{L}^{j_2}$ into the set of 2D observed line segment of $\mathbf{L}^{j_1}$. 3D endpoints are updated by computing the mean of back-projected 3D endpoint within the new set of 2D line segments.

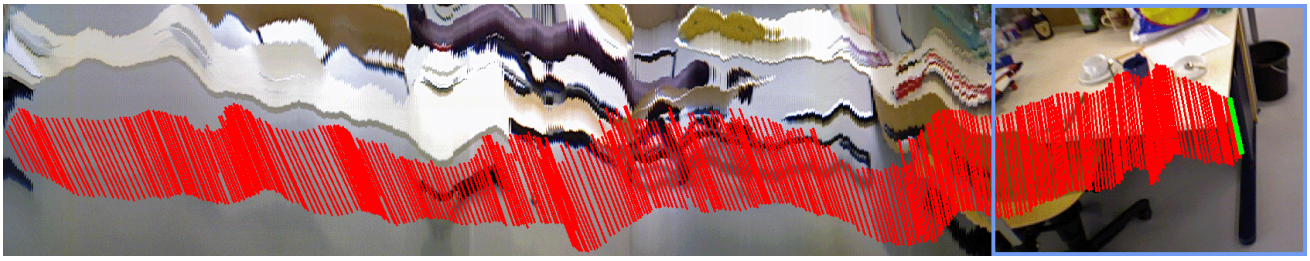


Fig. 8: Line flow visualization. We display the tracking result of a line flow over 300 frames on the *fr3_long_office* sequence. Green line segments are in current frame (blue) while red line segments are in previous ones (refer to the supplementary video for more information).

Local bundle adjustment. Local bundle adjustment is performed by minimizing the reprojection error of points and line segments:

$$\mathrm{E} = \min_{\{\mathbf{T}\},\{\mathbf{L}\},\{\mathbf{P}\}} \sum_{\{\mathbf{l}\},\{\mathbf{p}\}} \rho(\boldsymbol{\epsilon}_{\mathbf{l}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{l}} \boldsymbol{\epsilon}_{\mathbf{l}} + \boldsymbol{\epsilon}_{\mathbf{p}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{p}} \boldsymbol{\epsilon}_{\mathbf{p}}), \tag{16}$$

where we optimize the camera poses $\mathbf{T}_{\varphi}$, line segments $\mathbf{L}$ and points $\mathbf{P}$ in the covisibility graph. When we process the current keyframe, the camera poses $\mathbf{T}_{\varphi}$ include the current keyframe and its connected keyframes in the local map. Other keyframes are included in the optimization but remain fixed. Two keyframes are connected when they share sufficient 3D features. A local map of the processed keyframe collects the connected keyframes near the processed keyframe in 3D space.

We check the line reprojection errors within the line flow. When the number of outliers is larger than half of the total number of the line segments within the line flow, we delete the line flow. In addition, we re-compute the mean of 3D back-projected endpoints from 2D line segments to update 3D endpoints.

Loop closure. When we detect a loop closure using the method in [40], we update all the keyframes and the map with a similarity transformation to eliminate scale drifts. A two-step strategy is implemented to accelerate the convergence. First, following [40], we obtain observed pose-pose constraints with $\mathbf{T}_{i,j}^{o}$, $\mathbf{T}_{i,j} \in \mathbf{T}_{\phi}$ to perform a pose graph optimization:

$$\mathrm{E} = \min_{\mathbf{T}_{\phi}} \sum_{\mathbf{T}_{i,j} \in \mathbf{T}_{\phi}} \rho(\boldsymbol{\epsilon}_{\mathbf{T}_{i,j}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{T}_{i,j}} \boldsymbol{\epsilon}_{\mathbf{T}_{i,j}}), \tag{17}$$

$$\boldsymbol{\epsilon}_{\mathbf{T}_{i,j}} = \mathbf{T}_{i,j}^{o} \mathbf{T}_{j} \mathbf{T}_{i}^{-1}. \tag{18}$$

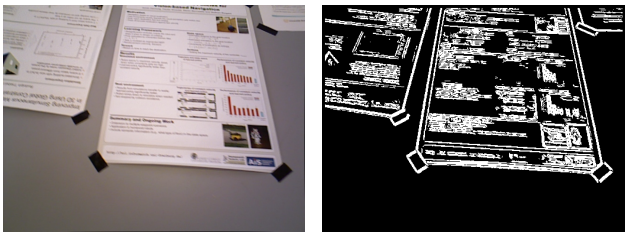
To deal with the problems brought by scale drifts, we transform the 3D line $\mathbf{L}$ from the world coordinate to the reference keyframe coordinates before optimization. Then, the parameters of all the keyframes and the 3D map are jointly optimized in the similarity transformation:

$$\mathrm{E} = \min_{\{\mathbf{T}\},\{\mathbf{L}\},\{\mathbf{P}\}} \sum_{\{\mathbf{l}\},\{\mathbf{p}\}} \rho({}_{s}\boldsymbol{\epsilon}_{\mathbf{l}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{l}s} \boldsymbol{\epsilon}_{\mathbf{l}} + {}_{s}\boldsymbol{\epsilon}_{\mathbf{p}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{p}s} \boldsymbol{\epsilon}_{\mathbf{p}}), \tag{19}$$

$${}_{s}\boldsymbol{\epsilon}_{\mathbf{l}} = \frac{1}{\sqrt{l_x^2 + l_y^2}} \left[\mathbf{e}_1 | \mathbf{e}_2\right]^{T} \mathbf{K}_{l} \left[\mathbf{R} | s \left[\mathbf{t}\right]_{\times} \mathbf{R}\right] \mathbf{L}, \tag{20}$$

$${}_{s}\boldsymbol{\epsilon}_{\mathbf{p}} = \mathbf{p} - \lfloor s \mathbf{K}_{p} (\mathbf{R}\mathbf{P} + \mathbf{t}) \rfloor. \tag{21}$$

After this optimization, a similar procedure is performed in the local bundle adjustment module to discard outliers.



(a) RBG image (b) Visited pixels

Fig. 9: Visualization of visited pixels by guided line detection in the *fr3_nostr_tex_near* sequence. (a) An 640 × 480 image; (b) shows The corresponding pixels (white) needed to be processed.

VI. Experiments

We compare our approach with state-of-the-art SLAM systems on indoor and outdoor datasets, including the TUM RGBD [41], the 7-Scenes [42], the EuROC [43] and the KITTI [44] datasets. We only use the monocular RGB images in these datasets. Both accuracy and efficiency are evaluated to demonstrate that we achieve promising accuracy compared with state-of-the-art SLAM methods. We also provide 3D line maps and line flow tracking results as qualitative evaluation to demonstrate that reliable and stable line flows can be extracted in challenging scenarios.

A. Implementation Details

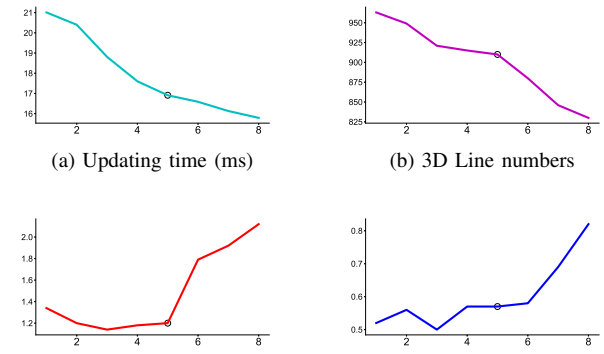
All the experiments are performed on a desktop PC with a 3.6GHz Core i7-7700 CPU and 16GB memory. We use the benchmark tool to align metrics and eliminate scale ambiguity before computing the error.

We utilize Levenberg Marquardt algorithm implemented in Ceres solver [45] for solving nonlinear least squares problems. In the appendix, we discuss line representation and the Jacobian matrices of error terms used in optimization procedure.

Point management. We extract ORB features in each image. Features are separated into equal cells by dividing the images. We calculate ORB descriptors for point matching. We estimate a robust essential matrix with RANSAC [35]. The 3D points are triangulated to check rotation and translation decomposed from the essential matrix. Initialization is achieved when the inlier number of 3D points is greater than a threshold. We initialize the succeeded frames as keyframes. For the following frames, a two-step strategy is performed to enhance the robustness and efficiency of our system. First, we leverage the predicted camera poses $\mathbf{T}_{t+1}$ as coarse poses. By leveraging

the coarse poses, we search 2D candidates near the 2D projected point for each 3D point. A grid strategy is applied for accelerating the procedure by dividing images into cells and keypoints are collected according to the corresponding cells. We optimize the predicted camera poses $\mathbf{T}_{t+1}$ with the point correspondences. Second, we search more correspondences by projecting 3D points from local maps. Then, these 2D-3D correspondences are applied in short term optimization. In long-term optimization, 3D points are triangulated when a new keyframe is created. We use geometric checking to find duplications. The management of point features is similar to ORB-SLAM2 [7]. Since the procedures for line and point features are independent, we manage them in parallel.

Parameter setting. The collinearity in terms of 2D and 3D is assessed when the angular difference between the two line segments is less than a threshold. Too strict parameters (2D: $< 3°$ and 3D: $< 5°$) can lead to the failure of collinearity check, and many broken line segments can be generated. The possible setting ranges are 2D: $3° \sim 10°$ and 3D: $5° \sim 15°$. Higher values generate more false positive segments. Considering the trade-off between recall and precision, we set the 2D line collinearity threshold to $5°$; and the 3D line collinearity threshold to $10°$. The number of holding frame α is set to $2 \sim 4$ to avoid unnecessary termination of a line flow when the line segment correspondences are not stable in a few frames. Very large α can lead to more false positives, especially when line segments do not appear in consecutive frames. We set α to 3 to get reasonable performance. The changing ratio of line length β is set to 0.8 to smooth the change of line length. Small values (≤ 0.7) decrease the motion constraint of a line segment, which make this operation not very useful. Large values (≥ 0.85) lead to many incorrect endpoints. The width of the rectangular search area illustrated in Fig. 6 (b) is set according to previous support region and 2D motions, and 5×5 seeds are uniformly sampled within the rectangle because the seeds nearby the area usually are similar. The grids $1 \times 1 \sim 4 \times 4$ output fewer line candidates, we may find incorrect candidates and line flow tracking fails. $5 \times 5 \sim 10 \times 10$ is a reasonable range for sufficient line candidates. For the size of the fitting window, it's sufficient to estimate camera motion by setting the size of window $t_w = 2$. However, the velocity estimation using only two frames can be influenced by noises. Therefore, we set $t_w = 5$ to obtain a more stable velocity. Window sizes ranging from $3 \sim 7$ bring similar performance in our experiments.



(a) Updating time (ms) (b) 3D Line numbers

(c) Mean translation error (cm/s) (d) Mean rotation error (deg/s)

Fig. 10: The influence of N_{cl} on updating time, 3D line number, mean translation error, and rotation error. The experiment is conducted on *fr3_long_office* sequence.

(a) 7-Scenes. *fire_02*

(b) 7-Scenes. *stairs_04*

(c) TUM. *fr2_desk_with_person*

Fig. 11: Line flow results in 3 challenging scenarios: partial occlusion (a), repetitive texture (b), and dynamic object occlusion (c). The left picture of each row shows the first line segment of the line flow (green), while the right one shows the current line segment of the line flow (green) along with previously extracted line segments (red).

B. Line Flow Analysis

N_{cl} determines the frequency of the running of a LSD for the remaining pixels in a frame after the pixels in the predicted regions are visited. N_{cl} is a very important parameter since it decides the efficiency and effectiveness of line flow extraction. We conduct a comprehensive study to understand the effect of N_{cl} on efficiency and robustness (shown in Fig. 10). A large N_{cl} leads to high efficiency. However, a very large N_{cl} increases pose errors. To balance the tradeoff between efficiency and robustness, we set N_{cl} to 5 in all the experiments.

Running time. To compare the efficiency of different line segment extraction and matching strategies, we first test the running time using LSD [10] for extraction and LBD [11] for matching. This strategy has been adopted by a few SLAM works [8], [9], [22]. It takes 37.97ms for each frame. As illustrated in Fig. 10 (a), setting N_{cl} to 1, the module can be viewed as LSD + line flow updating implementation and runs in 21.01ms. In comparison, our line flow predicting and updating implementation takes 16.91ms because we only process selected pixels. The guided line extraction and descriptor-

TABLE I
THE ABSOLUTE KEYFRAME TRAJECTORY ERROR (ATE) EVALUATION ON THE TUM RGBD BENCHMARK [41] (RMSE, cm). THE RESULTS OF PL-SLAM MONO (ORIGIN) AND LSD-SLAM ARE DERIVED FROM [9]. THE RESULT OF ORB-SLAM2 ARE DERIVED FROM [7]. THE RESULT OF DSO† [1] IS GENERATED FROM THE SOURCE CODE WITH THE DEFAULT PARAMETERS.

Method / Dataset	LF-SLAM	PL-SLAM Mono [9]	ORB-SLAM2 [7]	DSO† [1]	LSD-SLAM [2]
fr1_xyz	1.05	1.21	**0.90**	6.30	9.00
fr1_floor	**1.74**	7.59	2.99	5.25	38.07
fr2_xyz	**0.25**	0.43	0.30	0.98	2.15
fr2_360_kidnap	**2.97**	3.92	3.81	4.12	-
fr2_desk_with_person	**0.69**	1.99	0.88	-	31.73
fr3_str_tex_far	0.88	0.89	**0.77**	1.36	7.95
fr3_str_tex_near	**1.17**	1.25	1.58	7.26	-
fr3_nostr_tex_near	**1.36**	2.06	1.39	7.30	7.54
fr3_sit_halfsph	**1.29**	1.31	1.34	3.57	5.87
fr3_long_office	**1.35**	1.97	3.45	10.11	38.53
fr3_walk_xyz	**1.16**	1.54	1.24	14.14	12.44
fr3_walk_halfsph	1.66	**1.60**	1.74	31.86	-
average	**1.29**	2.15	1.70	8.39	17.03

TABLE II
THE AVERAGE RUNTIME OF EACH MODULE OF LF-SLAM (ms).

Resolution		TUM-RGBD 640 × 480	EuRoC MAV 752 × 480
Line Flow Tracking		14.29	19.93
Point Tracking		23.98	25.59
Short-term Opt.		3.69	3.01
Long-term Opt.	3D Line Proc.	0.83	1.74
	3D Point Proc.	25.55	23.30
	Local BA	257.80	301.28
	Loop Closure	1164.66	289.25

free matching ensure high efficiency. We illustrate the visited pixels of an images when we utilize the line flow updating strategy in Fig. 9. This strategy guarantees that line flow based SLAM runs in real-time ($25 - 35$ FPS). The timing of the different modules of LF-SLAM is shown in Table II. Line flow tracking is a joint procedure for feature extraction and 2D-2D matching. The module of line flow tracking runs at 70 FPS on the TUM-RGBD dataset and at 50 FPS on the EuRoC MAV, respectively, because of the high image resolution of the EuRoC MAV dataset. The point tracking and line flow tracking run in parallel. The front-end tracker (including line flow tracking, point tracking, and short-term optimization), takes 28 ms/frame on the TUM-RGBD and the EuRoC MAV datasets. We put the long-term optimization in another thread to achieve real time performance. Note that since 3D-2D line matching is automatically done in line flows based on the line flow definition. Line merging is very efficient because most of the correspondences for line segments are already found and kept in line flows.

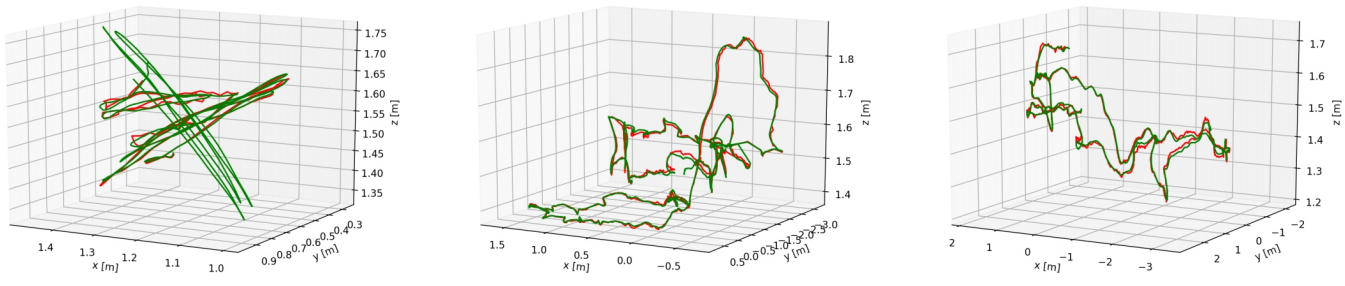
We visualize a line flow on the *fr3_long_office* sequence in Fig. 8. Starting from the first frame, we show all the 2D line segments. The line segments move with the camera motion. At the same time, the endpoints and line direction are stable on each frame from different views.

In addition, we present our line flow results in 3 typical scenarios. Fig. 11(a) demonstrates that we can track a line segment with stable endpoints during partial occlusions. Fig. 11 (b) visualizes that we preserve the consistency of line segments under such repetitive texture scenario. We achieve promising tracking results as illustrated in Table III. Fig. 11 (c) demonstrates that we extract a line segment through prediction guidance and fusion. Therefore, the edge of the table does not split, and we fuse them as a whole when partial occlusion induced by a dynamic object occurs.

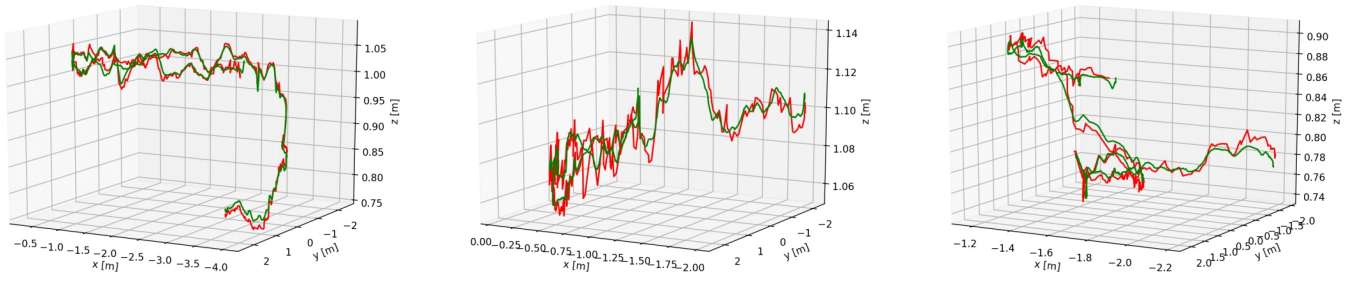
C. Quantitative Evaluation

Quantitative Evaluation Baselines. We evaluate LF-SLAM against a few state-of-the-art SLAM algorithms: ORB-SLAM2 [7], PL-SLAM [9], LSD-SLAM [2] and DSO [1]. DSO [1] and LSD [2] are direct methods that not only operate well in texture-less environments, but also provide visually appealing reconstruction results. Note that PL-SLAMs have monocular [9] and stereo [8] versions in the literature, we label the monocular version as PL-SLAM Mono [9]. Both ORB-SLAM2 [7] and PL-SLAM Mono [9] are indirect methods. ORB-SLAM2 has achieved state-of-the-art performance on many datasets. PL-SLAM Mono exploits point and line features to deal with low texture scenarios.

Experiments on the TUM RGBD benchmark. A quantitative evaluation of localization accuracy on the TUM RGBD benchmark is demonstrated in Table I. The TUM RGBD dataset consists of 39 indoor sequences captured in an office environment and an industrial hall. Indirect methods, ORB-SLAM2, PL-SLAM Mono, and LF-SLAM provide better performance than direct methods such as DSO and LSD-SLAM. Although direct methods can generate visually appealing dense 3D maps, it is not easy to leverage their global maps to improve localization accuracy. LF-SLAM achieves the best results on 9 of 12 sequences. The mean ATEs of our method compared against other methods prove that LF-SLAM system has good performance in different scenarios. We also give the trajectories provided by LF-SLAM and the ground truth trajectories in Fig. 12.



(a) *fr1_xyz* (b) *fr2_desk_with_person* (c) *fr3_long_office*



(d) *fr3_nostr_tex_near* (e) *fr3_str_tex_far* (f) *fr3_str_tex_near*

Fig. 12: The trajectories of LF-SLAM (red) and the ground truth (green) on TUM RGBD dataset. Our trajectories are very closed to the ground truth trajectories.

fr1_floor sequence contains a few knotholes, which can be easily tracked by point features. In contrast, salient line features rarely appear in this sequence. Due to this reason, the results of PL-SLAM Mono are inferior to those of ORB-SLAM2. By carefully maintaining line flows, LF-SLAM achieves the best results by utilizing the temporarily visible lines on objects and the repetitive textures on the floor.

fr3_str_tex_near sequence has abundant texture. However, frequent camera jitters at close range lead to motion blurs. Thanks to the reserving period setting, line flows can be kept stably during image blur. And our LF-SLAM system achieves the best result compared with other SLAM algorithms.

fr3_long_office contains desks and has a large loop closure. Both PL-SLAM Mono and LF-SLAM achieve much better results than that of ORB-SLAM2. This proves the advantage brought by the integration of line and point features.

Experiments on the 7-Scenes benchmark. We further evaluate our method on the 7-Scenes benchmark [41]. The sequences are captured by a handheld camera in indoor scenes in 7 different indoor environments with diverse sequences for each environment. As shown in Table III, our LF-SLAM outperforms other methods on 18 out of 21 sequences, and obtains the lowest average error among the three methods.

stairs sequence is quite challenging due to the repetitive structures of the stairs. ORB-SLAM2 fails on this sequence. In contrast, the proposed LF-SLAM successfully handles this challenging sequence with line flows.

pumpkin and *redkitchen* sequences both contain sufficient features with regular and constantly visible lines. Hence, the incorporation of point and line features brings better accuracy. In contrast, in *fire* sequences, stable lines can hardly be observed in the sequences. LF-SLAM can only depend on point features, and achieve similar results with ORB-SLAM2.

Experiments on the EuRoC MAV dataset. The EuRoC MAV dataset consists of 11 stereo-inertial sequences recorded in different indoor environments with structure information. These sequences were captured by randomly walking in the rooms. Therefore, loops with different sizes are recorded in these sequences. Table IV shows the RMSEs of the camera trajectory on the sequences with different motions. LF-SLAM achieves the best performance on 8 of 11 sequences. Due to the inability of the large size of the local maps and loop closure, the performance of DSO is worse than ORB-SLAM2 and LF-SLAM. The performance of LF-SLAM is better than ORB-SLAM2 because of the additional structural information. *V1-03-diff* includes long-term fast and abrupt motions. Because the long term stable features cannot be triangulated and maintained properly and loop detection fails, the result of DSO is better than ORB-SLAM2 and LF-SLAM on this sequence. In *V1-03-diff* and *MH-04-diff* sequences, the long term abrupt motion limits the ability of LF-SLAM to maintain stable lines, the RMSE of LF-SLAM is worse than ORB-SLAM2.

D. Qualitative Evaluation

To reconstruct a scene, we collect an image sequence in an office room with a monocular camera (Kinect V2). Fig. 13 demonstrates a few images captured in the rooms. Fig. 14

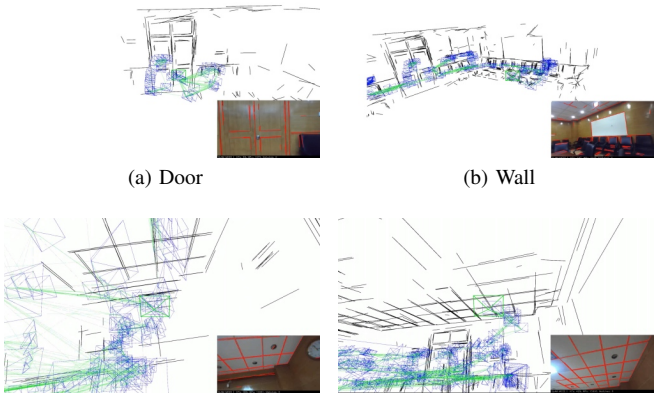
Fig. 13: Some images we captured in an office room. These images contains challenges such as texture-less regions, image blur, illumination variations, similar appearance and overexposure regions.

TABLE III
THE ATE EVALUATION ON THE 7-SCENES [42] (RMSE, cm). FOR EACH SCENE, FOUR SEQUENCES ARE USED FOR EVALUATION. "-" DENOTES THAT THE SYSTEM FAILS IN THE SEQUENCE. THE RESULTS OF ORB-SLAM† ARE GENERATED FROM THE OFFICIALLY RELEASED SOURCE CODE WITH TUNNED PARAMETERS.

Scene_no	LF-SLAM	ORB-SLAM2†
chess_01	**4.20**	5.06
chess_02	**4.21**	4.06
chess_03	**3.66**	7.74
chess_04	**5.01**	7.25
fire_01	**4.12**	4.94
fire_02	**2.46**	3.10
fire_03	4.14	**3.56**
fire_04	**4.43**	4.44
heads_01	**5.96**	6.66
heads_02	**3.66**	4.98
office_01	**9.38**	9.74
office_02	**9.21**	11.44
office_03	**7.95**	11.80
office_04	**11.71**	16.06
stairs_01	**12.61**	48.04
stairs_02	**9.10**	-
stairs_03	**5.06**	-
stairs_04	**6.42**	-
pumpkin_01	**9.91**	13.25
pumpkin_02	**2.03**	4.62
pumpkin_03	**8.03**	10.21
pumpkin_06	**8.31**	8.90
redkitchen_01	**3.92**	6.03
redkitchen_02	**13.04**	15.63
redkitchen_03	**4.95**	7.53
redkitchen_04	**3.37**	5.81
average	**6.41**	9.60

TABLE IV
THE COMPARISON RESULTS ON EUROC MAV DATASET (RMSE, cm).

Sequence	DSO [1]	ORB-SLAM2 [7]	LF-SLAM
V1-01-easy	15.9743	8.8047	**8.7703**
V1-02-med	49.9751	6.2277	**6.1678**
V1-03-diff	**115.6318**	117.2230	125.5764
V2-01-easy	6.9131	5.9825	**5.9040**
V2-02-med	9.8438	5.6164	**5.4818**
V2-03-diff	147.5033	11.8120	**9.7286**
MH-01-easy	13.1358	3.8071	**3.7232**
MH-02-easy	10.5858	**3.1169**	3.1949
MH-03-med	50.273	3.74965	**3.6450**
MH-04-diff	43.9722	**6.8078**	7.0441
MH-05-diff	32.8823	5.2553	**5.2242**



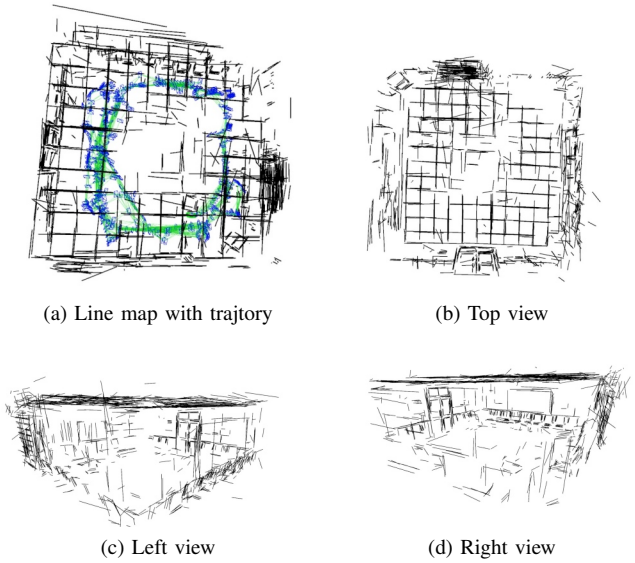
(a) Door (b) Wall

(c) Ceiling (d) Ceiling

Fig. 14: The incremental reconstruction. The current images are shown on bottom-right, while 3D line map and camera trajectory are shown on top-left. (a) Door recovery; (b) Wall Recovery; (c-d) Ceiling recovery.

visualizes the on-the-fly reconstruction of our algorithm. The red line segments on the bottom right image are the extracted line segments, while the dash lines are the reconstructed 3D line segments. Note that although a few line segments are not extracted in a single frames, and some line segments are incomplete, such line segments are refined based on the information in multiple frames. In Fig. 14 (a) to (d), we extract 2D line segments from the images successfully despite of the challenges such as similar textures, illumination variations, and reflect light.

We visualize the final 3D line map in Fig. 15. We recover most of the parts of the scenes in the office such as the ceiling, desk, white board. It also validates the good reconstruction ability of our method due to the reliable correspondences in the line flows.



(a) Line map with trajtory (b) Top view

(c) Left view (d) Right view

Fig. 15: Reconstructed 3D line map. (a) The 3D line map and the camera trajectory. (b-d) The different views of the 3D line map.

All these qualitative results verify our contribution on fully exploiting the spatial-temporal coherence of line features to maintain stable and reliable line flows, which is the key to accurate pose estimation and map reconstruction.

VII. Conclusions

We propose a novel line flow representation for describing line segments in consecutive frames. Line flows encode spatial-temporal coherence of line segments in image sequences by considering the correspondences among 2D and 3D line segments. Based on line flows, we developed LF-SLAM. LF-SLAM can deal with many challenging scenarios such as texture-less images, occlusions, image blur, and features with similar appearances. The efficiency of LF-SLAM is higher than other systems. Compared with other state-of-the-art direct and indirect methods, our system achieves good performance on four datasets. In addition, LF-SLAM generates precise and visually appealing 3D line maps on-the-fly.

Line flows maintain 2D line segments by fully exploiting the spatial-temporal constraints. Our system fails when the camera motion breaks the constraints, especially when long-term abrupt camera motion occurs. One possible solution to solve this problem is to adopt a coarse-to-fine strategy by extracting line segments from image pyramids. The pose estimation can be performed based on the image scales from small to large.

Currently, our line flows model the coherence of line segments in monocular sequences. In the future, we will extend the representation to SLAM systems with stereo and RGBD cameras. Moreover, we plan to compositely model flows of rich types of features, e.g., lines and planes. This is expected to enhance the robustness of SLAM systems to large camera motion, improve localization accuracy and generate more complete 3D maps.

Appendix A
Non-linear Optimizations

A. Line Representation for Optimization

We use an iterative method to efficiently minimize a non-linear energy function. The Plücker coordinates has 6 DOF variables with 2 strict constraints. In our optimization process, it is difficult for the iterative method to keep the strict constraints. To deal with this problem, we transform line representation from Plücker coordinates into orthonormal representation which has 4 DOF variables. We rearrange 3D line $\mathbf{L} = [\mathbf{n}|\mathbf{v}]$ and leverage QR decomposition to obtain orthonormal representation:

$$\begin{aligned}\mathbf{L} &= \sqrt{\|\mathbf{n}\|^2 + \|\mathbf{v}\|^2}\mathbf{U}\begin{bmatrix}\sigma_1 & \\ & \sigma_2\end{bmatrix} \\ &= \sqrt{\|\mathbf{n}\|^2 + \|\mathbf{v}\|^2} \\ &\quad \begin{bmatrix}\frac{\mathbf{n}}{\|\mathbf{n}\|} & \frac{\mathbf{v}}{\|\mathbf{v}\|} & \frac{\mathbf{n}\times\mathbf{v}}{\|\mathbf{n}\times\mathbf{v}\|}\end{bmatrix}\begin{bmatrix}\frac{\|\mathbf{n}\|}{\sqrt{\|\mathbf{n}\|^2+\|\mathbf{v}\|^2}} & \\ & \frac{\|\mathbf{v}\|}{\sqrt{\|\mathbf{n}\|^2+\|\mathbf{v}\|^2}}\end{bmatrix}, \\ \mathbf{W} &= \begin{bmatrix}\sigma_1 & -\sigma_2 \\ \sigma_2 & \sigma_1\end{bmatrix},\end{aligned} \tag{22}$$

here, $\mathbf{U} \in \mathrm{SO}(3)$ and $\mathbf{W} \in \mathrm{SO}(2)$. The orthonormal representation $(\mathbf{U}, \mathbf{W}) \in \mathrm{SO}(3) \times \mathrm{SO}(2)$ corresponds to a unique 3D line. We refer reader to [30] for more details.

B. Jacobian Matrices of Error Terms

we deduce point and line errors term of Jacobian matrices using left version definition [47]. We utilize the Lie algebra $\mathfrak{se}(3)$ of the corresponding rigid transformation $\mathbf{T}$ to minimize the energy function. We derived point error term $\boldsymbol{\epsilon}_p$ with respect to 3D point $\mathbf{J}_{\mathbf{P}}^{\boldsymbol{\epsilon}_\mathbf{P}}$ and camera pose $\mathbf{J}_{\mathbf{T}}^{\boldsymbol{\epsilon}_\mathbf{P}}$, and line error term $\boldsymbol{\epsilon}_l$ with respect to 3D line $\mathbf{J}_{\mathbf{l}}^{\boldsymbol{\epsilon}_\mathbf{l}}$ and camera pose $\mathbf{J}_{\mathbf{T}}^{\boldsymbol{\epsilon}_\mathbf{l}}$. We have:

$$\begin{aligned}\mathbf{J}_{\mathbf{P}}^{\boldsymbol{\epsilon}_\mathbf{P}} &= \mathbf{J}_{\mathbf{p}}^{\circ}\mathbf{R}, \\ \mathbf{J}_{\mathbf{T}}^{\boldsymbol{\epsilon}_\mathbf{P}} &= \mathbf{J}_{\mathbf{p}}^{\circ}\left[[-\mathbf{R}\mathbf{P}]_\times|\mathbf{I}\right], \\ \mathbf{J}_{\mathbf{L}}^{\boldsymbol{\epsilon}_\mathbf{l}} &= \mathbf{J}_{\mathbf{L}}^{\circ}\left[\mathbf{R}|[\mathbf{t}]_\times\mathbf{R}\right]\begin{bmatrix}\mathbf{0}_{3\times1} & -\sigma_1\mathbf{u}_3 & \sigma_1\mathbf{u}_2 & -\sigma_2\mathbf{u}_1 \\ \sigma_2\mathbf{u}_3 & \mathbf{0}_{3\times1} & -\sigma_2\mathbf{u}_1 & \sigma_1\mathbf{u}_2\end{bmatrix}, \\ \mathbf{J}_{\mathbf{T}}^{\boldsymbol{\epsilon}_\mathbf{l}} &= \mathbf{J}_{\mathbf{L}}^{\circ}\left[-[\mathbf{R}\mathbf{n}]_\times - [\mathbf{t}]_\times[\mathbf{R}\mathbf{v}]_\times| - [\mathbf{R}\mathbf{v}]_\times\right],\end{aligned} \tag{23}$$

where, $\mathbf{u}_i$ is the i-th column of $\mathbf{U}$,

$$\begin{aligned}\mathbf{J}_{\mathbf{p}}^{\circ} &= \begin{bmatrix}\frac{1}{p_z} & 0 & \frac{-p_x}{p_z^2} \\ 0 & \frac{1}{p_z} & \frac{-p_y}{p_z^2}\end{bmatrix}\mathbf{K}_p, \\ \mathbf{J}_{\mathbf{L}}^{\circ} &= (l_x^2 + l_y^2)^{-3/2}\left[\mathbf{e_1}|\mathbf{e_2}\right]^T\begin{bmatrix}l_y^2 & -l_xl_y & 0 \\ -l_xl_y & l_x^2 & 0 \\ -l_xl_z & -l_yl_z & l_x^2+l_y^2\end{bmatrix}\mathbf{K}_l,\end{aligned} \tag{24}$$

here, (p_x, p_y, p_z) and (l_x, l_y, l_z) are the homogeneous coordinate of projected 3D point and 3D line, respectively.

References

[1] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 40, no. 3, pp. 611–625, 2018.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale Direct Monocular SLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849.

[3] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[4] K. Joo, T.-H. Oh, I. So Kweon, and J.-C. Bazin, "Globally Optimal Inlier Set Maximization for Atlanta Frame Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5726–5734.

[5] H. Li, J. Yao, J.-c. Bazin, X. Lu, Y. Xing, and K. Liu, "A Monocular SLAM System Leveraging Structural Regularity in Manhattan World," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2518–2525.

[6] 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 (VT)*, vol. 64, no. 4, pp. 1364–1375, 2015.

[7] 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 (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] R. Gomez-Ojeda, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A Stereo SLAM System through the Combination of Points and Line Segments," *IEEE Transactions on Robotics (T-RO)*, vol. 35, no. 3, pp. 734–746, 2017.

[9] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time Monocular Visual SLAM with Points and Lines," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4503–4508.

[10] 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 (T-PAMI)*, vol. 32, no. 4, pp. 722–732, 2010.

[11] 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 (VCIR)*, vol. 24, no. 7, pp. 794–805, 2013.

[12] E. J. Almazan, R. Tal, Y. Qian, and J. H. Elder, "MCMLSD: A Dynamic Programming Approach to Line Segment Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5854–5862.

[13] N. Cho, A. Yuille, and S. Lee, "A Novel Linelet-based Representation for Line Segment Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 40, no. 5, pp. 1195–1208, 2018.

[14] D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition (PR)*, vol. 13, no. 2, pp. 111–122, 1981.

[15] C. Akinlar and C. Topal, "EDLines: A Real-time Line Segment Detector with a False Detection Control," *Pattern Recognition Letters (PRL)*, vol. 32, no. 13, pp. 1633–1642, 2011.

[16] Y. Salaun, R. Marlet, and P. Monasse, "Multiscale Line Segment Detector for Robust and Accurate SfM," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2000–2005.

[17] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, "Learning Attraction Field Representation for Robust Line Segment Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1595–1603.

[18] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7105–7114.

[19] Z. Wang, H. Liu, and F. Wu, "HLD: A Robust Descriptor for Line Matching," in *Proceedings of International Conference on Computer-Aided Design and Computer Graphics (CADCG)*, 2009, pp. 128–133.

[20] B. Micusik and H. Wildenauer, "Structure from Motion with Line Segments under Relaxed Endpoint Constraints," *International Journal of Computer Vision (IJCV)*, vol. 124, no. 1, pp. 65–79, 2017.

[21] B. Fan, F. Wu, and Z. Hu, "Robust Line Matching through Line-point Invariants," *Pattern Recognition (PR)*, vol. 45, no. 2, pp. 794–805, 2012.

[22] Y. Zhao and P. A. Vela, "Good Line Cutting: Towards Accurate Pose Tracking of Line-assisted VO/VSLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 527–543.

[23] E. Perdices, L. M. López, and J. M. Cañas, "LineSLAM: Visual Real Time Localization using Lines and UKF," in *Proceedings of First Iberian Robotics Conference - Advances in Robotics*, 2013, pp. 663–678.

[24] J. Solà, 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 (IJCV)*, vol. 97, no. 3, pp. 339–368, 2012.

[25] 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 (T-RO)*, vol. 31, no. 6, pp. 1364–1377, 2015.

[26] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust Visual SLAM with Point and Line Features," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)* , 2017.

[27] Y. Lu and D. Song, "Robust RGB-D Odometry using Point and Line Features," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015, pp. 3934–3942.

[28] D. Ruifang, V. Frémont, S. Lacroix, I. Fantoni, D. Ruifang, V. Frémont, S. Lacroix, I. Fantoni, and L. C. L.-b. Monocu, "Line-based Monocular Graph SLAM," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2017, pp. 494–500.

[29] N. Ayache and B. Faverjon, "Efficient Registration of Stereo Images by Matching Graph Descriptions of Edge Segments," *Proceedings of International Journal of Computer Vision (IJCV)*, vol. 1, no. 2, pp. 107–131, 1987.

[30] A. Bartoli and P. Sturm, "Structure-from-Motion using Lines: Representation, Triangulation, and Bundle Adjustment," *Computer Vision and Image Understanding (CVIU)*, vol. 100, no. 3, pp. 416–441, 2005.

[31] A. Jain, C. Kurz, T. Thormahlen, and H. Seidel, "Exploiting Global Connectivity Constraints for Reconstruction of 3D Line Segments from Images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1586–1593.

[32] S. He, X. Qin, Z. Zhang, and M. Jagersand, "Incremental 3D Line Segment Extraction from Semi-dense SLAM," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1658–1663.

[33] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3D Scene Abstraction using Line Segments," *Computer Vision and Image Understanding (CVIU)*, vol. 157, pp. 167–178, 2017.

[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 (VCIR)*, vol. 25, no. 5, pp. 904–915, 2014.

[35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[36] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transactions on Robotics (T-RO)*, vol. 24, no. 6, pp. 1365–1378, 2008.

[37] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, 2005.

[38] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A Line Segment Detector," *Image Processing On Line (IPOL)*, vol. 2, pp. 35–55, 2012.

[39] Jianbo Shi and Tomasi, "Good Features to Track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600.

[40] R. Mur-Artal and J. D. Tardos, "Fast Relocalisation and Loop Closing in Keyframe-based SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 846–853.

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[42] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2930–2937.

[43] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC Micro Aerial Vehicle Datasets," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, pp. 1157–1163, 2016.

[44] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[45] S. Agarwal, K. Mierle, and Others, "Ceres Solver," http://ceres-solver.org, 2012.

[46] J. L. Schonberger and J. Frahm, "Structure-from-Motion Revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[47] J. Sol, J. Deray, and D. Atchuthan, "A Micro Lie Theory for State Estimation in Robotics," *arXiv:1812.01537v6*, 2018.