Structure PLP-SLAM: Efficient Sparse Mapping and Localization using Point, Line and Plane for Monocular, RGB-D and Stereo Cameras

Fangwen Shu Jiaxuan Wang Alain Pagani Didier Stricker

Abstract— **This paper presents a visual SLAM system that uses both points and lines for robust camera localization, and simultaneously performs a piece-wise planar reconstruction (PPR) of the environment to provide a structural map in real-time. One of the biggest challenges in parallel tracking and mapping with a monocular camera is to keep the scale consistent when reconstructing the geometric primitives. This further introduces difficulties in graph optimization of the bundle adjustment (BA) step. We solve these problems by proposing several run-time optimizations on the reconstructed lines and planes. Our system is able to run with depth and stereo sensors in addition to the monocular setting. Our proposed SLAM tightly incorporates the semantic and geometric features to boost both frontend pose tracking and backend map optimization. We evaluate our system exhaustively on various datasets, and show that we outperform state-of-the-art methods in terms of trajectory precision. The code of PLP-SLAM has been made available in open-source for the research community (https://github.com/PeterFWS/Structure-PLP-SLAM).**

I. Introduction

In human-made environments, surrounding structures can often be represented by planes and line segments. Leveraging those higher-level features in visual SLAM systems has the potential to improve camera localization performance and to generate structural maps instead of unstructured point clouds. While most of the existing systems are based only on feature points and use sparse points clouds to describe the scenes and estimate the camera poses [6], [7], [9], [21], [33], these methods face various challenges in practical application, such as lacking points in low-texture environments, or poor matching performance in changing light conditions.

A line segment is a geometric primitive that has dual relation with the point, and can therefore be used in SLAM systems as efficiently as points. PL-SLAM systems [13], [35] were proposed using points and lines following the pipeline of ORB-SLAM2 [33]. However, these systems adopt a representation of a 3D line with two endpoints, which can lead to ambiguities on the line direction and thus on line description (see Fig. 2 (a)) and suffer from occlusions or misdetections. In an attempt to solve this problem, mono PL-SLAM [35] followed the idea of EPnPL [43] by forcing the endpoint correspondences with a two-step procedure when minimizing the reprojection error. However, this two-step procedure is not compatible with optimizations based on graph optimization frameworks (e.g. g2o [22]). In recent works such as Structure-SLAM [27], the optimization of the reprojection error on 3D lines is operated by optimizing two 3D endpoints, a representation that is over-parameterized. Since this over-parametrization cannot reliably reconstruct 3D lines, it is a factor of deficiency in the system. To overcome these problems, in our framework, we represent 3D lines using Plücker coordinates which can be converted to an orthonormal representation with minimal parameterization [3]. In order to simplify the map visualization, 3D-2D correspondences search, and graph optimization, we maintain both endpoints and Plücker coordinates, and use different representations for different tasks.

Planes, especially for indoor environments, are predominant structures that are less affected by measurement noise, as planes can be extracted e.g. from the depth image [54]. Hence, planes are often used in RGB-D SLAM [16], [17], [19], [20], [24] with an ICP-like registration and optimization, or constraints such as Manhattan World (MW) [5], [8], [51]. Another benefit of using planar structures is that large areas can be mapped very efficiently and used as a semantic indicator to enable intuitive and useful AR applications [36]. However, very limited works employ monocular SLAM as the backbone when considering planes due to scale ambiguity and the difficulty of fitting planes without depth sensors. In our case, following the work of Pop-up SLAM [48] which segments the ground plane using a neural network, we use an instance planar segmentation CNN [29], [45], [46] to generate a plane prior, then we incrementally reconstruct and refine the 3D piece-wise planar structure using directly the sparse point cloud generated by the monocular SLAM.

The contributions of our structure PLP-SLAM algorithm are threefold: (1) We first propose a modularized multi-feature monocular SLAM system that exploits line detection, tracking and mapping, real-time piece-wise planar reconstruction, and joint graph optimization in addition to the standard feature points. (2) We show that the loop closure can be efficiently done with a corrected line map and propose a re-localization module based on the pre-built point-line map. (3) We propose an extension of our system for RGB-D and stereo cameras. This makes our framework versatile and sensor-agnostic while tightly incorporating semantic features in SLAM. In addition, our SLAM framework is designed to be robust to noisy input in order to reduce the dependency on predictions of the CNN-based semantic planar detection. In practice, the environment observed by the camera might be very diverse, and therefore the lines and planes are reconstructed and optimized without the need of any strong assumption such as MW. Since points, lines and planes are tightly integrated as basic features, our system is not limited

DFKI - German Research Center for Artificial Intelligence. E-mail: {first_name}.{last_name}@dfki.de. **Acknowledgment:** This research has been partially funded by the German BMBF project MOVEON (01IS20077) and EU project CORTEX2 (grant agreement: N° 101070192).

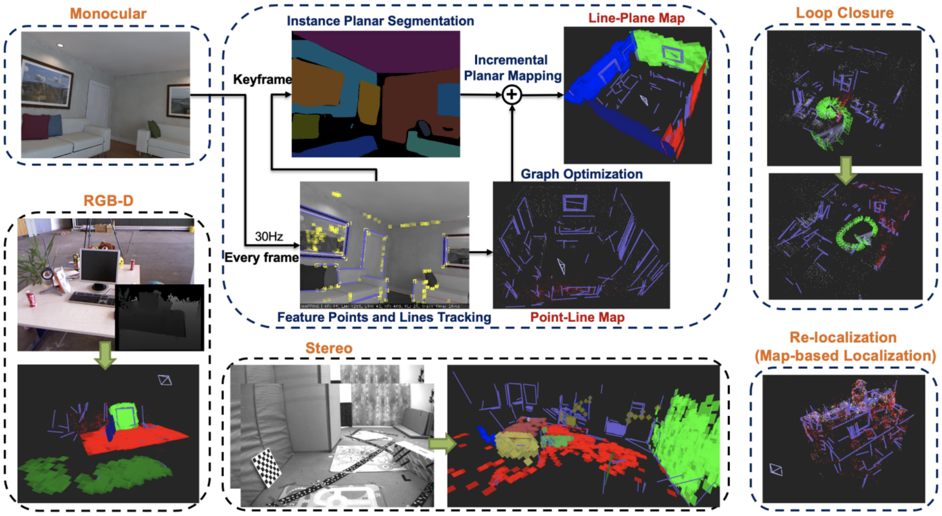


Fig. 1: **The proposed SLAM system** utilizes point and line cloud for robust camera localization, with piece-wise planar reconstruction for semantic mapping. Here, the reconstructed maps are presented selectively for better visualization.

to small-scale scenes. We benchmark our SLAM exhaustively on indoor datasets TUM RGB-D [40], ICL-NUIM [14], and EuRoC MAV [4] showing superior quantitative results compared to other state-of-the-art SLAM systems. The workflow of our SLAM framework is illustrated in Fig. 1, qualitatively showing that our reconstructed maps are accurate, intuitive, and semantically meaningful.

II. Method

In this section, we describe our monocular SLAM system built upon OpenVSLAM [42] which is a derivative of ORB-SLAM2 [33] with high usability and extensibility. Then we highlight our contributions and the added modules.

A. *Structure-based Monocular SLAM*

One of the central modules of a SLAM system is the joint optimization of the map and the camera poses in bundle adjustment (BA). Since we are aiming at integrating higher-level features in addition to feature points, we need a proper and efficient representation of 3D lines as well as the corresponding Jacobian matrix for iterative optimization. We integrate the planes, by adding a constraint only between 3D points and 3D planes, because the definition of a reprojection error for planar primitives is ambiguous due to their invariance to certain translations and rotations.

1) ***Exploiting Line Segments:*** We extract 2D line segments using LSD [44] and match them across frames via LBD descriptors [53]. The parameters of LSD are optimized for the best trade-off between computation efficiency and accuracy via a hidden parameter tuning and length rejection strategy, as reported in PL-VINS [10]. Following the same idea, we achieve line segment extraction which is 3 times faster than the original implementation from OpenCV. We then use two types of representations for the line segments:

Simple Representation of Two Ending Points. For the visualization of a 3D line segment in the map, we use the two endpoints. This brings the advantage of a trivial projection of the 3D endpoints on the image plane, which can be used for fast retrieval of 3D-2D line matches using LBD descriptors. It also allows us to track partially occluded lines, even when one the of endpoints falls outside of the image frustum. It is important to note that the choice of the 3D endpoints has no influence on the non-linear optimization since we use another 3D line representation for the BA, as introduced in the following.

Plücker Coordinates and Orthonormal Representation. A 3D line can be represented with the 6D vector of Plücker coordinates as $\mathbf{L} = (\mathbf{m}^\top, \mathbf{d}^\top)^\top$, where the vector $\mathbf{m} \in \mathbb{R}^3$ (also called as the moment vector) is the normal to the interpretation plane containing the line $\mathbf{L}$, and $\mathbf{d} \in \mathbb{R}^3$ indicates the line direction, see Fig. 2 (b). Notice that it is an infinite 3D line representation, where $\mathbf{m}$ and $\mathbf{d}$ do not need to be unit vectors in the implementation. In this way, a 3D line can be transformed from world coordinates to camera coordinates via a transformation matrix similar to 3D points, and then be projected on the image plane:

$$\mathbf{L}_c = \begin{bmatrix} \mathbf{m}_c \\ \mathbf{d}_c \end{bmatrix} = \mathbf{T}_{cw}\mathbf{L}_w = \begin{bmatrix} \mathbf{R}_{cw} & [\mathbf{t}_{cw}]_\times \mathbf{R}_{cw} \\ \mathbf{0} & \mathbf{R}_{cw} \end{bmatrix} \begin{bmatrix} \mathbf{m}_w \\ \mathbf{d}_w \end{bmatrix} \quad (1)$$

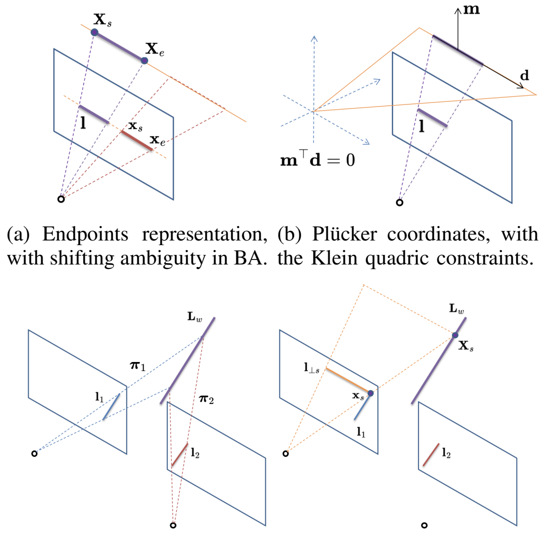
$$\mathbf{l} = [l_1, l_2, l_3]^\top = \mathbf{K}_L \mathbf{m}_c, \quad (2)$$

where the $\mathbf{R}_{cw} \in SO(3)$ and $\mathbf{t}_{cw} \in \mathbb{R}^3$ are the standard rotation and translation of the camera pose. The $\mathbf{K}_L$ is the intrinsic matrix used to project 3D line on the image plane:

$$\mathbf{K}_L = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y c_x & -f_x c_y & f_x f_y \end{bmatrix}$$

Hence, we are able to define the reprojection error of the 3D line $\mathbf{L}_w$ with its 2D correspondence in the image:

$$\mathbf{e}_\mathbf{l} = \left[\frac{\mathbf{x}_s{}^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}}, \frac{\mathbf{x}_e{}^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \right]^\top \quad (3)$$



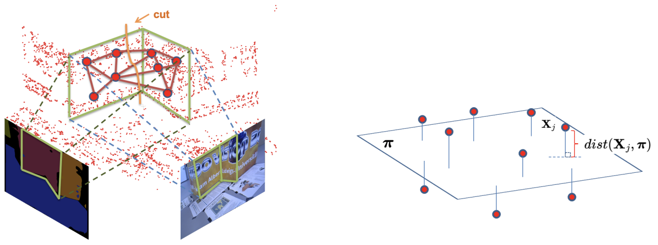
(a) Endpoints representation, with shifting ambiguity in BA.

(b) Plücker coordinates, with the Klein quadric constraints.

(c) Two-view triangulation.

(d) Endpoints trimming.

Fig. 2: **The 3D line** representations and reconstruction.



(a) 3D plane reconstruction with spatial coherence.

(b) The possible distribution of 3D map points around a plane [23].

Fig. 3: **The interpretation of 3D plane:** (a) An example of 3D plane fitting optimized with spatial coherence. We initialize the plane by the given instance planar segmentation. Note that if two planes are wrongly segmented as one (e.g. due to texture similarity), they are separated later thanks to our graph-cut within a local neighborhood graph in 3D, as highlighted. (b) The possible distribution of the 3D points.

where $\{\mathbf{x}_s, \mathbf{x}_e\}$ are the 2D starting point and ending point of an extracted line segment from LSD.

This error term is used in BA for optimizing 3D lines and camera poses. However, Plücker coordinates are still over-parameterized, as it presents 3D line with 5-DOF in homogeneous coordinates satisfying the Klein quadric constraints $\mathbf{m}^\top\mathbf{d} = 0$ [15]. For updating the Plücker coordinates during the iterative optimization, we convert them to the minimal 4-DOF orthonormal representation [3], and convert back after the optimization step. The conversion between orthonormal representation and Plücker coordinates is given in [10], [52]. We implement the bundle adjustment using g2o [22].

Two-view Triangulation of Lines. Reconstructing a line $\mathbf{L}_w$ in 3D can be achieved by forward projecting matched 2D line segments $\mathbf{l}_1$ and $\mathbf{l}_2$ from two image views to give two 3D planes, and intersecting these two planes:

$$\boldsymbol{\pi}_1 = \mathbf{l}_1^\top\mathbf{P}_1, \boldsymbol{\pi}_2 = \mathbf{l}_2^\top\mathbf{P}_2 \tag{4}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are the standard 3×4 camera projection matrices. 2D lines are constructed via their two endpoints $\{\mathbf{x}_s, \mathbf{x}_e\}$ using cross product: $\mathbf{l} = \mathbf{x}_s \times \mathbf{x}_e$. After that, the Plücker coordinates $(\mathbf{m}^\top, \mathbf{d}^\top)^\top$ can be extracted from the dual Plücker matrix $\mathbf{L}^*$ [15], which has the properties of:

$$\mathbf{L}^* = \boldsymbol{\pi}_1\boldsymbol{\pi}_2^\top - \boldsymbol{\pi}_2\boldsymbol{\pi}_1^\top = \begin{bmatrix} [\mathbf{d}]_\times & \mathbf{m} \\ -\mathbf{m}^\top & \mathbf{0} \end{bmatrix} \tag{5}$$

However, the triangulated 3D line is an infinite line from two intersected 3D planes, see Fig. 2 (c). In order to estimate the 3D endpoints for visualization and matching, using the method of endpoints trimming as discussed below.

Endpoints Trimming and Outlier Rejection. The endpoints of a 3D line can be estimated using the corresponding 2D line segment from its reference keyframe, as illustrated in Fig. 2 (d). Endpoints trimming was introduced in [25], [52] as a standalone method only for visualization. In this work, we further utilize the endpoints trimming within the iterative local BA for outlier rejection and map culling, in addition to the χ^2 distribution test on the reprojection error. It is integrated as part of the positive depth checking (such as $Z_c > 0$), and evaluates if the absolute change of the position ΔX of the 3D endpoints compared to the median depth of the scene is less than a ratio (0.1) after optimization finished. If it is not the case, the 3D line is an outlier (possibly from mismatching or triangulated with two-view ambiguity when the 3D line is located close to the epipolar plane). Experimentally, we found that this makes the BA more efficient and robust to outliers, the reconstructed line cloud remains accurate, compact, and clean, which can be observed from Fig. 1 and Fig. 4. Moreover, endpoints trimming is also used to correct the line map during the loop closure (see Sec. II-C).

*2) **Exploiting 3D Planar Structures**:* The reconstruction of 3D planes in a monocular SLAM is challenging because of the limited 3D information. At the same time, the Manhattan World (MW) assumption does not necessarily hold if we are trying to reconstruct all the possible plane instances observed from a single image (not limited to the perpendicular layout). In order to reconstruct planes, we leverage the pairwise relationship between the 3D points and 3D planes in SLAM backend, and utilize the geometric relationship to minimize the 3D distance, as illustrated in Fig. 3 (b). Here, we adopt the infinite plane representation $\boldsymbol{\pi} = (\mathbf{n}^\top, d)^\top$ [19], where $\mathbf{n}$ is the plane normal and d is the distance to the world origin.

A 3D plane instance is reconstructed by fitting a set of sparse and noisy 3D points triangulated in real-time. We initialize the set of points belonging to a single plane by predicting the instance planar segmentation of the input images using PlaneRecNet [46] (only on keyframes). However, in order to cope with possible misclassification from the neural network, especially on the unseen data of SLAM benchmarks, the reconstruction is conducted as a sequential RANSAC [38] coupled with an inner local optimization of Graph-cut [1]. In this way, we locally optimize the spatial coherent planes in 3D space, as shown in Fig. 3 (a). To this aim, we formulate the plane fitting problem as an optimal binary labeling problem [18] with the energy term:

$$E(\mathbf{\Pi}) = \sum_{v} \|\Pi_v\| + \lambda \cdot \sum_{(u,v)\in\mathcal{N}} \delta(\Pi_u \neq \Pi_v) \tag{6}$$

where the first term $\sum_v \|\Pi_v\|$ counts inliers for the target plane model using 0-1 measurement:

$$\|\Pi_v\|_{\{0;1\}} = \begin{cases} 0 & \text{if } (\Pi_v = 1 \wedge \operatorname{dist}(\mathbf{v},\boldsymbol{\pi}) < \epsilon_d) \vee \\ & \quad (\Pi_v = 0 \wedge \operatorname{dist}(\mathbf{v},\boldsymbol{\pi}) \geq \epsilon_d) \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

In Eq. (6), $\mathbf{\Pi} = \{\Pi | \mathbf{v} \in V\}$ is the assignment of plane models to 3D point $\mathbf{v}$, and V indicates the set of 3D vertices from a neighborhood graph. Here, we use the distance between a 3D point and the plane as the geometric error measure in Eq. (7): $\operatorname{dist}(\mathbf{v},\boldsymbol{\pi}) = |\frac{\mathbf{n}^\top\mathbf{v}+d}{\|\mathbf{n}\|}|$.

Moreover, in Eq. (7) the parameter $\Pi_v \in \{0, 1\}$ indicates the labeling. Here, the unary energy penalizes nothing when a 3D point is labeled as an inlier (close to the plane) or it is labeled as an outlier (far from the plane). The second term of Eq. (6) indicates the spatial regularization [18] which penalizes neighbors with different labels in the graph. $\delta(\cdot)$ is 1 if the specified condition inside the parenthesis holds, and 0 otherwise. The neighborhood graph $\mathcal{N}$ is constructed using the Fast Approximate Nearest Neighbors algorithm [30] according to a predefined sphere radius r $(= 2\epsilon_d)$, and the minimum samples (3 points formulate a plane) are sampled uniformly. λ is a parameter balancing the two terms, which is set as 0.6 in our experiments.

Incremental Merging and Refinement. In order to cope with possibly large planes, we add a merging mechanism in the local mapping thread. Two planes are merged if the following two conditions are met: first they should have nearly parallel normals: $|cos(\theta)| = |\frac{\mathbf{n_i n_j}}{\|\mathbf{n_i}\|\|\mathbf{n_j}\|}| > T_\theta$ (set as 0.8 in this work) and second, they should be geometrically close to each other: $|\frac{d_i}{\|d_i\|} - \frac{d_j}{\|d_j\|}| < T_d$. The new plane equation is then updated according to a model residual threshold ϵ_Π in a RANSAC loop. Following this, all associated point landmarks are projected on the plane by minimizing the point-plane distance via: $\hat{\mathbf{v}} = \mathbf{v} - \operatorname{dist}(\mathbf{v},\boldsymbol{\pi})\frac{\mathbf{n}}{\|\mathbf{n}\|}$.

Adaptive Geometric Thresholds. Some of the parameters introduced in the previous equations need to be adjusted according to different environments. Examples of such parameters are ϵ_d in Eq. (7), the parameter T_d used to merge planes, and the model residual ϵ_Π used to stop the RANSAC loop. In order to avoid case-dependent parameter-tuning, we follow the work of [38] with an adaptive parameter setting strategy, where the above-mentioned thresholds will be adjusted dynamically according to the scene depth of the local map observed by the reference keyframe.

B. Motion-only BA and Local BA

In this work, the reprojection errors of the points and lines are minimized in two different bundle adjustments: motion-only BA and local BA. The overall cost function is:

$$C = \sum_{i,j} \rho_h(\mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij}) + \sum_{i,z} \rho_h(\mathbf{e}_{\mathbf{l}iz}^\top \mathbf{\Omega}_{iz} \mathbf{e}_{\mathbf{l}iz}) \tag{8}$$

implicitly with the minimized distance between 3D point and 3D plane, because the planes are statistically fitted in the optimal position via SVD, the associated 3D points are projected on the plane as discussed in the last Sec. II-A.2:

$$\sum_{j,k} dist(\mathbf{X}_j, \boldsymbol{\pi}_k)$$

where i, j, k, z are the number of camera views, 3D points, 3D planes, and 3D lines, respectively. In Eq. (8), the first term indicates the standard reprojection error for feature points, and the second term indicates the line reprojection error explained by Eq. (3). Moreover, the 6-DOF camera pose is represented as Lie algebra $\mathfrak{se}(3)$, and the 4-DOF line is represented as orthonormal representation. ρ_h is the Huber robust cost function and $\mathbf{\Omega}_{ij}, \mathbf{\Omega}_{iz}$ are the covariance matrices associated with the scale (of image pyramid) at which feature point or line segment was detected. In this work, we only utilize the line segments extracted from the original image resolution, thus $\mathbf{\Omega}_{iz} = \mathbf{I}_{2\times2}$.

The analytical Jacobians for point are well-known, while the Jacobians for line can be analytically calculated by chain rule to make derivations which are given in [3], [25], [55].

C. Loop Detection, Loop Closure and Global BA

Loop detection of monocular SLAM aims at estimating the 7-DOF similarity transformation $\mathrm{Sim}(3)$ after a best-validated loop candidate (keyframe) is found:

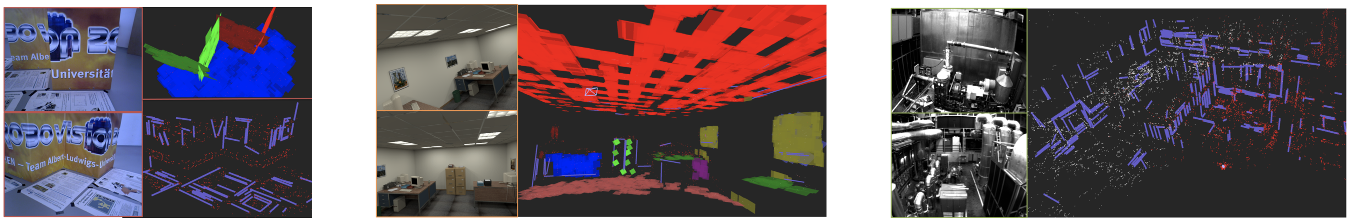
$$\mathrm{Sim}(3) = \left\{ \mathbf{S}_{\text{point}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{4\times4} \right\} \tag{9}$$

In this work, we do not address the place recognition problem by re-building a BoW (Bag of Words) vocabulary using the LBD descriptors as done in [13]. Instead, we use the given DBoW vocabulary [11] built from ORB features [32] for loop detection. The 3D line similarity transformation [2] is then calculated according to Eq. (9):

$$\mathbf{S}_{\text{line}} = \begin{bmatrix} s\mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{10}$$

where both similarity transformations are expressed by the same scale factor, rotation and translation matrices.

Thus, we are able to correct the 3D line map (represented by the Plücker coordinates) in a similar way as the correction of a 3D point cloud within the step of Loop Fusion described in [31]. Thereafter, we optimize the Essential graph over the similarity transformations, which distributes the loop closing error along the graph and correct scale drifts [39]. Then the 3D map points and lines are transformed according to the correction of the reference keyframe that observed them. This procedure is illustrated in Fig. 1. To achieve the optimal solution, we perform a global BA in a separate thread [33], and endpoints trimming is used to re-estimate 3D line endpoints and function as a map culling method.



(a) (**Monocular**) Map of *fr3_structure_texture_far*. This example shows that our line and plane reconstruction are accurate when using monocular SLAM.

(b) (**RGB-D**) Map of *office_room_traj0*. Here, the ceiling (red) is perfectly reconstructed, and the furniture can be observed combined with the plane (blue/green) and lines. Same for the paintwork (yellow) on the wall. This example verifies that our sparse semantic mapping is efficient and intuitive.

(c) (**Stereo**) Point-line map of *MH_04_difficult*. No planar structure due to the failure of CNN on this factory sequence. However, our map is as good as the map reconstructed from PL-VINS [10] which is a visual-inertial SLAM using points and lines.

Fig. 4: **The qualitative results of various reconstructed maps** under different sensor settings, complementary to Fig. 1.

Monocular	**Ours** (full)	**Ours**	**Ours**	Structure SLAM [27]	Object- Plane [47]
Config.	point + line + plane	point + line	point + plane	point + line + plane + normal	point + plane + object
living_rm_0	**0.26**	0.32	0.39	-	0.80
living_rm_2	2.21	**1.88**	2.67	4.50	2.06
living_rm_3	1.95	**1.81**	2.54	4.60	5.38
office_0	5.14	**4.50**	5.26	-	5.93
office_2	4.07	3.55	5.31	3.10	**2.63**
office_3	3.67	**2.95**	5.65	6.50	-
Avg.	2.88	**2.50**	3.64	4.68	3.36

TABLE I: **Monocular SLAM** evaluated on **ICL-NUIM [14]**, presented are the **absolute trajectory errors (ATEs) RMSE [cm]** (- stands result not available). Each result from ours was calculated as the average over 5 executions.

Monocular	**Ours** (full)	**Ours**	**Ours**	PL-SLAM [35]	Elaborate [25]	Structure SLAM [27]
Config.	point + line + plane	point + line	point + plane	point + line	point + line	point + line + plane + normal
fr1_xyz	1.06	1.05	1.09	1.21	**1.02**	-
fr1_floor	2.03	2.24	**1.85**	7.59	3.49	-
fr1_desk	2.02	**1.65**	1.82	-	-	-
fr2_xyz	0.26	0.26	**0.25**	0.43	0.31	-
fr2_desk	0.94	**0.77**	1.14	-	4.65	-
fr3_st_tex_far	0.99	1.11	1.05	0.89	**0.87**	1.40
fr3_st_tex_near	**1.14**	1.44	1.15	1.25	-	1.40
fr3_nst_tex_near	**1.26**	1.41	1.48	2.06	-	-
fr3_nst_tex_far	**3.24**	3.37	3.51	-	3.68	-
fr3_long_office	1.16	**1.04**	1.54	1.97	2.98	-
Avg.	**1.41**	1.43	1.49	2.20	2.43	-

TABLE II: **Monocular SLAM** evaluated on dataset **TUM RGB-D** [41], presented are the **ATEs RMSE [cm]**.

D. Re-localization

The existing method in feature-based SLAM utilizes the global descriptor of BoW [11] for image retrieval, thereafter using the $O(n)$ closed-form solution of EPnP [26] to initialize the iterative optimization, as the run-time requirement is critical. Therefore, simply replacing the EPnP with EPnPL [43] as done in mono PL-SLAM [35] brings no significant improvement. In this work, we use a BA with both point and line reprojection errors that provide better-refined camera poses. Note that we optimize over orthonormal representation of the line, instead of forcing the endpoints correspondence as done in [35] (in the spirit of EPnPL), which means that our method is naturally more efficient and avoids the shifting ambiguity of the line during nonlinear optimization.

III. Experiments and Results

We report our experiments on the datasets TUM RGB-D [40] and ICL-NUIM [14] for monocular and RGB-D SLAM. We only present the qualitative evaluation on EuRoC MAV [4] due to limited pages, please refer to Fig. 1 and Fig. 4.

A. Performance of Visual SLAM System

Effectiveness of line segments. We present the trajectory errors of our monocular SLAM compared to other state-of-the-art systems. The evaluated dataset ICL-NUIM provides low-contrast and low-texture synthetic indoor image sequences which are very challenging for monocular SLAM, while the TUM RGB-D dataset provides real-world indoor sequences under different texture and structure conditions. The effectiveness of adding line segments with a well-formulated reprojection error in SLAM can be found in the 3rd column (**ours**: point + line) of Table I. We observe that line segments improve the performance of monocular SLAM remarkably in the case of lacking feature points under low-texture and low-contrast environments. It also brings SLAM a more robust camera localization performance when there are enough point features tracked (on TUM RGB-D dataset of Table II), and similar results can be observed from Table III and Table IV with RGB-D setting.

The implicit constraint from 3D planar structures. In Table II, we observe that our implicit constraint that minimizes the distance between the 3D points and planes brings performance improvement to monocular SLAM on pure planar scenes such as *fr1_floor*, *fr3_structure_texture_near* and *fr3_nostructure_texture_far*, *fr3_nostructure_texture_near_with_loop*. Moreover, in Table III where we evaluate our RGB-D SLAM, when the depth sensor is available, this simple and efficient constraint also regularizes the point cloud (associated with certain planes) triangulated from the depth image, which results in the best average performance (**ours**: point + line + plane). This point-plane distance constraint is theoretically possible to be integrated into graph optimization [23] with a unary edge linked to the 3D point vertex, while the position of the plane is considered statistically optimal from RANSAC and SVD, hence fixed during optimization. Practically, this is equivalent to what we implemented via a simple projection of the 3D point on the plane. We experimentally found out

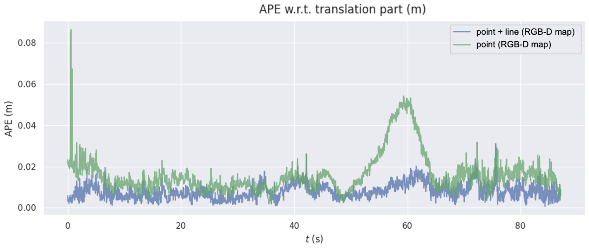


Fig. 5: **The evaluation of re-localization module.** We illustrate the absolute pose error (APE) [m] of all the relocalized images from sequence *freiburg3_long_office_household*, where the x-axis indicates the relative timestamps.

that adding this constraint into the pose-graph will disturb the optimization procedure, because the local BA then becomes a large-scale non-linear optimization problem since such unary edges do not directly constrain the camera poses, and certain planes (e.g. floor) may dominate the map and violates the local optimization strategy.

Failure cases. We observed that, e.g., monocular SLAM is not able to initialize on sequence *living_room_1* (of ICL-NUIM) which can be solved by adjusting the initialization threshold. On sequence *office_room_1* the monocular SLAM will lose its tracking due to brutal movement when the camera targets a no-texture corner of the room. This happens to all other monocular SLAM systems, so we removed these two sequences from evaluation for consistency (in Table I). Another factor of failure in our framework is semantic planar mapping, which will be discussed in the next section.

B. Sparse Semantic Mapping

In Fig. 1 and 4, we illustrate reconstructed point-line and line-plane maps respectively. Our map representation is designed as a lightweight sparse map, which intuitively shows the scene structure. The 3D planar structure is visualized using a rectangular plane-patch centered around the associated 3D point. In this way, we make use of the non-structure point cloud for visualizing the structural plane, efficiently without added computation. 3D lines are naturally observable, which illustrates most edge features of the scene.

3D plane reconstruction may fail, when the CNN-based segmentation is too noisy or even fails on ambiguous images. Thus, we use a graph-based optimization in RANSAC to fit planes. However, it is not avoidable that some user-defined thresholds of RANSAC are difficult to fine-tune. We solve this partially by introducing our adaptive parameter setting strategy [38]. One drawback of our planar map representation is that it strongly depends on the map points, which brings the limitation that no reliable plane can be fitted when there are not enough point landmarks. Our map refiner only keeps high-quality planes, which also omits smaller planes from the map. This results in mapping failure for low-texture scenes. Notice that the failure of planar mapping will not negatively influence the tracking thread of SLAM using points and lines, however, we remove the results from ours in Table IV if the plane reconstruction failed.

RGB-D	**Ours** (full)	**Ours**	**Ours**	Manhattan SLAM [51]	Structural RGB-D [28]	SP-SLAM [54]
Config.	point + line + plane	point + line	point + plane	point + line + plane	point + line + plane	point + plane
living_rm_0	**0.54**	0.63	0.58	0.70	0.60	0.80
living_rm_2	**1.26**	1.46	1.36	1.50	2.00	1.92
living_rm_3	**0.68**	1.07	1.16	1.10	1.20	1.25
office_0	**1.91**	1.99	2.31	2.50	4.10	1.99
office_1	2.66	2.26	2.20	**1.30**	2.00	2.25
office_2	0.83	**0.79**	0.89	1.50	1.10	2.20
office_3	**0.84**	0.96	0.86	1.30	1.40	1.84
Avg.	**1.25**	1.31	1.34	1.41	1.77	1.75

TABLE III: **RGB-D SLAM** evaluated on dataset **ICL-NUIM [14]**, presented are the **ATEs RMSE [cm]**.

RGB-D	**Ours** (full)	**Ours**	**Ours**	Manhattan SLAM[51]	SP-SLAM [54]
Config.	point + line + plane	point + line	point + plane	point + line + plane	point + plane
fr1_xyz	1.03	0.98	1.10	1.00	**0.93**
fr1_floor	**1.21**	1.40	1.36	-	-
fr1_desk	2.02	1.89	1.87	2.70	**1.43**
fr2_xyz	1.45	1.68	1.42	**0.80**	-
fr3_st_tex_far	**0.89**	0.99	0.98	2.20	0.97
fr3_st_tex_near	1.00	1.05	0.95	1.20	**0.84**
fr3_nst_tex_near	1.59	**1.19**	1.64	-	-
fr3_nst_tex_far	3.52	**3.05**	4.35	-	-
fr3_long_office	1.01	**0.91**	0.94	-	-
fr3_large_cabinet	4.99	4.37	5.03	8.30	**2.97**
fr3_st_notex_far	-	**1.46**	-	4.00	2.20
fr3_st_notex_near	-	**0.98**	-	2.30	1.25
Avg.	1.87	1.66	1.96	2.81	**1.51**

TABLE IV: **RGB-D SLAM** evaluated on dataset **TUM RGB-D** [41], presented are the **ATEs RMSE [cm]**.

C. Re-localization Module

Our re-localization module estimates the camera pose without using any prior information other than the map. This is useful when the previous camera location cannot be used, such as when the tracking is lost. This problem was recently addressed as the map-based visual localization problem [12], [49], [50] or global localization [34], [37]. We evaluate our re-localization module via conducting the single monocular image map-based localization, which is based on a pre-built map from our RGB-D SLAM system (as shown in Fig. 1 bottom-right, of sequence *freiburg3_long_office_household*). Thus, given every single image from this data sequence, we estimate the initial camera pose via image retrieval followed by EPnP, and then optimize it using motion-only BA with 3D-2D point and line correspondences, as discussed in Sec. II-D. To this end, we calculate the absolute camera position errors (APEs) using all relocalized images. As shown in Fig. 5, where the blue line indicates using points and lines for pose optimization, which shows clearly smaller errors compared to the one (green line) using points only.

IV. Conclusion

In this work, we presented a sparse visual SLAM method that uses both points and line segments for robust camera localization, together with a plane detection and reconstruction mechanism which increases both pose robustness and map semantic interpretation. Our comprehensive evaluations show high qualitative results of the reconstructed maps, and better quantitative performance in terms of pose estimation compared to state-of-the-art SLAM algorithms. As the system was designed based on monocular assumption, future work would be generalizing this full SLAM system to more challenging scenarios, such as scenes with low textures. We also plan to extend the system to other high-level features such as 3D objects in the scene.

References

[1] D. Barath and J. Matas. Graph-cut ransac: Local optimization on spatially coherent structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[2] A. Bartoli and P. Sturm. The 3d line motion matrix and alignment of line reconstructions. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[3] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer vision and image understanding*, 100(3):416–441, 2005.

[4] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[5] A. Concha, M. W. Hussain, L. Montano, and J. Civera. Manhattan and piecewise-planar constraints for dense monocular mapping. In *Robotics: Science and systems*, 2014.

[6] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[7] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[8] A. Flint, D. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *2011 International Conference on Computer Vision*, pages 2228–2235. IEEE, 2011.

[9] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[10] Q. Fu, J. Wang, H. Yu, I. Ali, F. Guo, Y. He, and H. Zhang. Pl-vins: Real-time monocular visual-inertial slam with point and line features, 2020.

[11] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[12] S. Gao, J. Wan, Y. Ping, X. Zhang, S. Dong, J. Li, and Y. Guo. Pose refinement with joint optimization of visual points and lines. *arXiv preprint arXiv:2110.03940*, 2021.

[13] R. Gomez-Ojeda, F.-A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez. Pl-slam: A stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35(3):734–746, 2019.

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

[15] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[16] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid. Structure aware slam using quadrics and planes. In *Asian Conference on Computer Vision*, pages 410–426. Springer, 2018.

[17] M. Hosseinzadeh, Y. Latif, and I. Reid. Sparse point-plane slam. In *Australasian Conference on Robotics and Automation*, 2017.

[18] H. Isack and Y. Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[19] M. Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[20] P. Kim, B. Coltin, and H. Jin Kim. Linear rgb-d slam for planar environments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 333–348, 2018.

[21] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[22] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g 2 o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

[23] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Mav visual slam with plane constraint. In *2011 IEEE International Conference on Robotics and Automation*, pages 3139–3144. IEEE, 2011.

[24] J.-K. Lee, J. Yea, M.-G. Park, and K.-J. Yoon. Joint layout estimation and global multi-view registration for indoor reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 162–171, 2017.

[25] S. J. Lee and S. S. Hwang. Elaborate monocular point and line slam with robust initialization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1121–1129, 2019.

[26] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009.

[27] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari. Structure-slam: Low-drift monocular slam in indoor environments. *IEEE Robotics and Automation Letters*, 5(4):6583–6590, 2020.

[28] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari. Rgb-d slam with structural regularities. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11581–11587. IEEE, 2021.

[29] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[30] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

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

[33] 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*, 33(5):1255–1262, 2017.

[34] S. Peng, Z. He, H. Zhang, R. Yan, C. Wang, Q. Zhu, and X. Liu. Megloc: A robust and accurate visual localization pipeline. *arXiv preprint arXiv:2111.13063*, 2021.

[35] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4503–4508. IEEE, 2017.

[36] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison. Dense planar slam. In *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 157–164. IEEE, 2014.

[37] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.

[38] F. Shu, Y. Xie, J. Rambach, A. Pagani, and D. Stricker. Visual slam with graph-cut optimized multi-plane reconstruction. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 165–170. IEEE, 2021.

[39] H. Strasdat, J. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2(3):7, 2010.

[40] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[42] S. Sumikura, M. Shibuya, and K. Sakurada. Openvslam: a versatile visual slam framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2292–2295, 2019.

[43] A. Vakhitov, J. Funke, and F. Moreno-Noguer. Accurate and linear time pose estimation from points and lines. In *European Conference on Computer Vision*, pages 583–599. Springer, 2016.

[44] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: a line segment detector. *Image Processing On Line*, 2:35–55, 2012.

[45] Y. Xie, J. Rambach, F. Shu, and D. Stricker. Planesegnet: Fast and robust plane estimation using a single-stage instance segmentation cnn. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13574–13580. IEEE, 2021.

[46] Y. Xie, F. Shu, J. Rambach, A. Pagani, and D. Stricker. Planerecnet: Multi-task learning with cross-task consistency for piece-wise plane detection and reconstruction from a single rgb image. *arXiv preprint arXiv:2110.11219*, 2021.

[47] S. Yang and S. Scherer. Monocular object and plane slam in structured environments. *IEEE Robotics and Automation Letters*, 4(4):3145–3152, 2019.

[48] S. Yang, Y. Song, M. Kaess, and S. Scherer. Pop-up slam: Semantic monocular plane slam for low-texture environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1222–1229. IEEE, 2016.

[49] H. Yu, W. Zhen, W. Yang, and S. Scherer. Line-based 2-d–3-d

registration and camera localization in structured environments. *IEEE Transactions on Instrumentation and Measurement*, 69(11):8962–8972, 2020.

[50] H. Yu, W. Zhen, W. Yang, J. Zhang, and S. Scherer. Monocular camera localization in prior lidar maps with 2d-3d line correspondences. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4588–4594. IEEE, 2020.

[51] R. Yunus, Y. Li, and F. Tombari. Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6687–6693. IEEE, 2021.

[52] 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*, 31(6):1364–1377, 2015.

[53] 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*, 24(7):794–805, 2013.

[54] X. Zhang, W. Wang, X. Qi, Z. Liao, and R. Wei. Point-plane slam using supposed planes for indoor environments. *Sensors*, 19(17):3795, 2019.

[55] X. Zuo, X. Xie, Y. Liu, and G. Huang. Robust visual slam with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1775–1782. IEEE, 2017.