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 demonstrates a visual SLAM system that utilizes point and line cloud for robust camera localization, simultaneously, with an embedded piece-wise planar reconstruction (PPR) module which in all provides a structural map. To build a scale consistent map in parallel with tracking, such as employing a single camera brings the challenge of reconstructing geometric primitives with scale ambiguity, and further introduces the difficulty in graph optimization of bundle adjustment (BA). We address these problems by proposing several run-time optimizations on the reconstructed lines and planes. The system is then extended with depth and stereo sensors based on the design of the monocular framework. The results show that our proposed SLAM tightly incorporates the semantic features to boost both frontend tracking as well as backend optimization. We evaluate our system exhaustively on various datasets, and open-source our code for the community (https://github.com/PeterFWS/Structure-PLP-SLAM).**

I. Introduction

In man-made environments, the geometric structures are often represented by line segments and planes. Using those higher-level features during mapping has the potential to improve camera localization performance and give substantial perception capabilities to visual SLAM systems. While most of the existing systems are based only on feature points and use sparse point cloud to describe the scenes and estimate the camera poses [6], [7], [9], [21], [33], these methods encounter various problems in practical application, such as low-texture environments, changing light, and motion-blur.

The line segment is a geometric primitive that has dual relation with the point, thus it produces valuable information as important as the point. PL-SLAM systems [13], [35] were proposed using points and lines following the pipeline of ORB-SLAM2 [31], [33]. However, they adopt 3D line with its two endpoints, which has shifting ambiguity on the line direction (Fig. 2 (a)) and subjects to occlusions or misdetections. For that reason, [35] intended to follow the idea of EPnPL [43] by forcing the endpoint correspondences with a two-step procedure when minimizing the reprojection error, but we found out this idea is not trivial to replicate into the graph optimization framework (e.g. g2o [22]). We also found out that those most recent works such as Structure-SLAM [27] follow the design of mono PL-SLAM [35], but the optimization of reprojection error on 3D line is operated as optimizing two 3D endpoints that are over-parameterized. This cannot reliably reconstruct 3D lines, which causes a factor of deficiency in the system. In our framework, we utilize the 3D lines using the well-studied Plücker coordinates which can be converted to the orthonormal representation of minimal parameterization. Later on we show that both endpoints and Plücker coordinates should be maintained, which simplifies the map visualization, fast 3D-2D correspondences searching, and graph optimization.

Planes, especially for indoor environments, are predominant structures that are less affected by measurement noise, as planes can be extracted from the depth image [54]. Hence, it is heavily used in RGB-D SLAM [16], [17], [19], [20], [24] with an ICP-like registration and optimization, with constraints such as Manhattan World (MW) [5], [8], [51]. Large areas can also be mapped efficiently and with the useful semantic planar structure which enables intuitive and useful AR applications [36]. However, very limited works employ monocular SLAM as backbone 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 take an instance planar segmentation CNN [29], [45], [46] for providing the prior, then we incrementally reconstruct and refine the piece-wise planar structure from the sparse point cloud triangulated along with the monocular SLAM.

In summary, we first contribute a modularized monocular SLAM system that exploits line tracking and mapping, real-time piece-wise planar reconstruction, and joint graph optimization in addition to standard feature points. Meanwhile, we show that the loop closure can be done with line map corrected, and a re-localization module based on the pre-built point-line map. The proposed workflow is then extended with RGB-D and stereo cameras. In this way, we build our structure-based sparse visual SLAM - a versatile framework that tightly incorporates semantic features. It is designed to be robust to noisy input such as the trained CNN may produce noisy predictions on unseen images, considering the added feature lines introduce more uncertainties and slows down the tracking as well as the optimization. The environment observed by the camera is in practical very diverse, thus lines and planes are reconstructed and optimized without forcing any strong assumption such as MW, when the points are still the basic geometric primitive, thus our system is not limited to small-scale scenes. We benchmark our SLAM exhaustively on indoor datasets TUM RGB-D [40], ICL-NUIM [14], and EuRoC MAV [4]. Competitive results as well as some superior quantitative results are presented in this work compared to other state-of-the-art SLAM systems. The workflow is illustrated in Fig. 1, qualitatively showing our reconstructed map is accurate, intuitive and efficient.

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 projects MOVEON (01IS20077) and SocialWear (01IW20002).

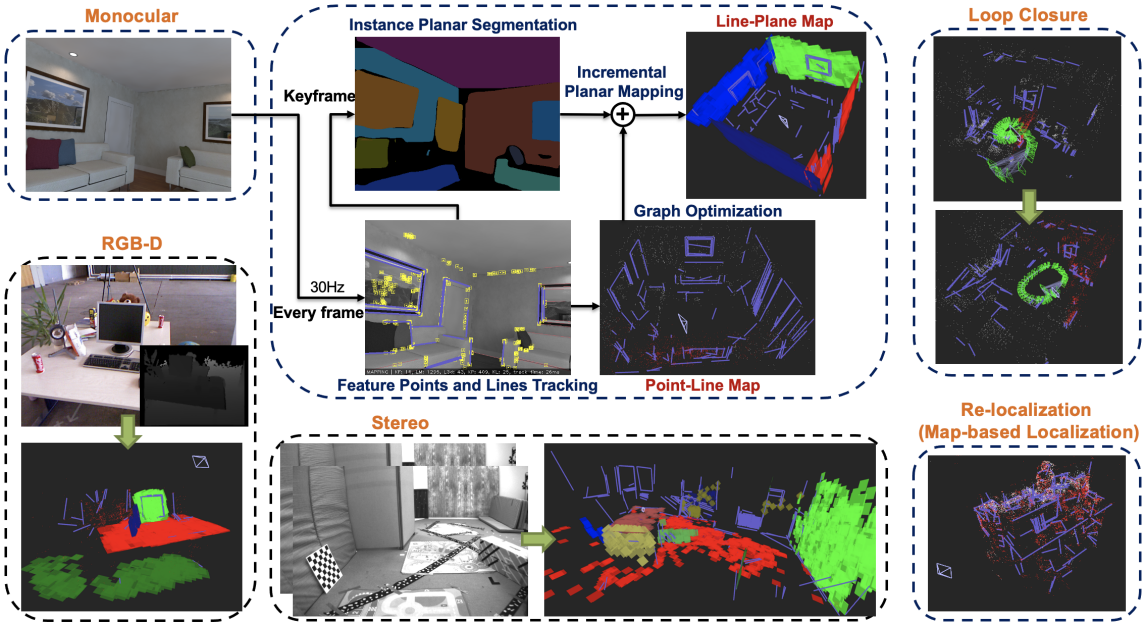


Fig. 1: **The proposed SLAM system** utilizes point and line cloud for robust camera localization, with piece-wise planar reconstruction for semantic mapping. Here, the map are presented selectively for better visualization.

II. Method

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

A. Structure-based Monocular SLAM

The integration of high-level features in addition to feature points requires a proper design of the Frame-to-Model strategy, where the camera poses and the map are iteratively optimized in BA. Thus, a proper representation of the 3D line is required, and corresponding Jacobian matrix (avoid numerical approximation and improve efficiency) used in the iterative optimization. In monocular SLAM, however, it is not trivial to define a reprojection error for plane primitives, thus we add constrain between 3D point and 3D plane.

*1) **Exploiting Line Segments**:* The 2D line segments are extracted by LSD [44] and matched across frames via LBD descriptor [53]. The parameters of the LSD are tweaked 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 way, we achieve line segments extraction which is 3 times faster than the original implementation from OpenCV.

Simple Representation of Two Ending Points. The 3D line in the map are visualized with its two endpoints. The advantage would be a trivial projection of 3D point on the image plane can be used for fast retrieval of 3D-2D line matches using LBD descriptors. This also allows us to track partially occluded lines, even when one the of endpoint falls outside of the image frustum. More important, which two 3D endpoints used for visualization have no influence on the non-linear optimization when we utilize the infinite 3D line representation, 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 this way, a 3D line can be transformed from world coordinates to camera coordinates via a transformation matrix similarly done to the 3D point, and then be projected on the image plane:

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

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

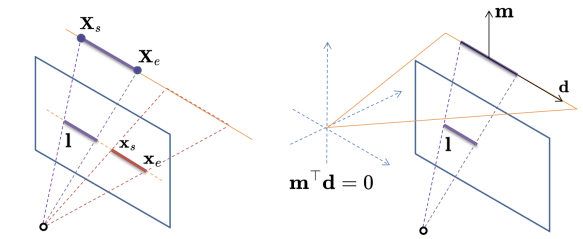
where the $\mathbf{R}_{cw} \in SO(3)$ and $\mathbf{t}_{cw} \in \mathbb{R}^3$ are the standard rotation and translation of the camera pose. The $\mathbf{K}_L$ is the intrinsic matrix used to project 3D line on the image plane:

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

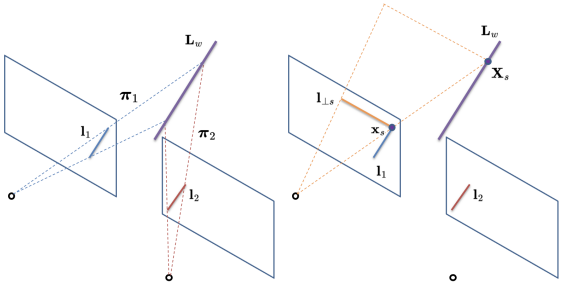
Hence, we are able to define the reprojection error of the 3D line $\mathbf{L}_w$ with its 2D correspondence:

$$\mathbf{e}_\mathbf{l} = \begin{bmatrix} \dfrac{\mathbf{x}_s{}^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}}, \dfrac{\mathbf{x}_e{}^\top \mathbf{l}}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix}^\top \quad (3)$$

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

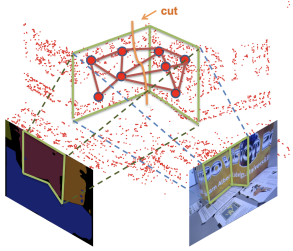


(a) Endpoints representation, with shifting ambiguity. (b) Plücker coordinates.

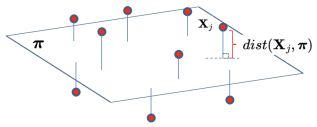


(c) Two-view triangulation. (d) Endpoints trimming.

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



(a) 3D plane reconstruction with spatial coherence.



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

Fig. 3: **The interpretation of 3D plane:** (a) An example that 3D plane fitting optimized with spatial coherence. It is initialized by the given instance planar segmentation, notice that there are two planes were wrongly segmented as one (often because of similar texture), which can be separated by graph-cut within a local neighborhood graph in 3D, as highlighted. (b) The possible distribution of the 3D points.

This error term will be 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 Plücker coordinates during iterative optimization, it will be converted to the minimal 4-DOF orthonormal representation [3], and converted back after optimization finished. The conversion between orthonormal representation and Plücker coordinates is given in [10], [52]. Our implementation is achieved 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 (Fig. 2 (c)), whose 3D endpoints have to be estimated for visualization, using the method of endpoints trimming as discussed in the following.

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 evaluate 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. Otherwise, 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 out 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 4. Moreover, endpoints trimming is also used to correct line map during the loop closure (see Sec. II-C).

*2) **Exploiting 3D Planar Structures**:* It is not trivial to reconstruct 3D planes in monocular SLAM because only limited 3D information can be obtained, and it is difficult to force the Manhattan World (MW) assumption if we are trying to reconstruct all the possible plane instances observed from a single image (not limited to the perpendicular layout).

Plane Interpretation. We seek to interpret the pairwise relationship between the 3D points and 3D planes in SLAM backend, and utilize the geometric relationship to minimize the 3D distance in-between, as illustrated in Fig. 3 (b). Here, with slightly abuse of the symbol, 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, which is initialized by the instance planar segmentation using [46] (only on keyframes). However, in order to cope with possible misclassification from the neural network, especially on the unseen dataset 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 explained by Fig. 3 (a). Thus, the plane fitting problem is formulated 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:

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

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): $\text{dist}(\mathbf{v},\boldsymbol{\pi}) = |\frac{\mathbf{n}^\top\mathbf{v}+d}{\|\mathbf{n}\|}|$.

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 this work.

Incremental Merging and Refinement. In the local mapping thread, we are trying to merge the closed planes (possibly also the smaller ones). Two planes are merged if the following two conditions are met, first they should have nearly parallel normals: $|cos(\theta)| = |\frac{\mathbf{n_i}\mathbf{n_j}}{\|\mathbf{n_i}\|\|\mathbf{n_j}\|}| > T_\theta$ (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. After that, all associated point landmarks will be projected on the plane by minimizing the point-plane distance via: $\hat{\mathbf{v}} = \mathbf{v} - \text{dist}(\mathbf{v},\boldsymbol{\pi})\frac{\mathbf{n}}{\|\mathbf{n}\|}$.

Geometric Thresholds. The user-defined parameters needs to be adjusted according to different environments, such as in Eq. (7) how to set ϵ_d in monocular SLAM. Similarly, for the parameter T_d used to merge planes, and the model residual ϵ_Π used to stop the RANSAC loop. For this reason, 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 motion-only BA and local BA, hence, 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, 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 original image resolution, thus $\mathbf{\Omega}_{iz} = \mathbf{I}_{2\times 2}$.

The analytical jacobians for point is 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 the 7-DOF similarity transformation $\text{Sim}(3)$ after a best validated loop candidate (keyframe) is found:

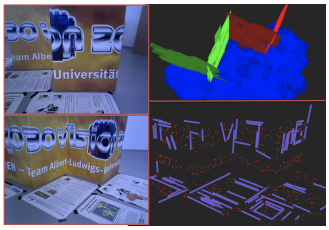
$$\text{Sim}(3) = \left\{ \mathbf{S}_{\text{point}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{4\times 4} \right\} \tag{9}$$

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

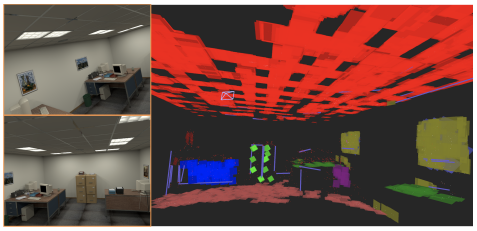
$$\mathbf{S}_{\text{line}} = \begin{bmatrix} s\mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{10}$$

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

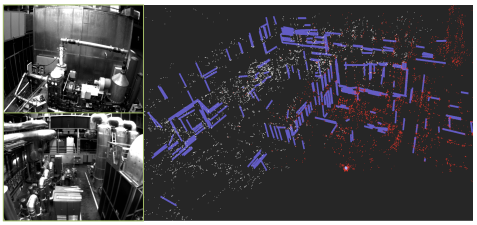
Thus, we are able to correct the 3D line map (represented by the Plücker coordinates) the same way as correcting 3D point cloud within the step of Loop Fusion described in [31]. After that, the Essential graph is optimized over the similarity transformations, which distributes the loop closing error 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, a global BA is suggested to be conducted in a separate thread [33]. Meanwhile, endpoints trimming is used to re-estimate 3D line endpoints after optimization, and function as a map culling method during BA.



(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 sequences. 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 dataset **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]	Elabroate [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. Thus, simply replacing the EPnP with EPnPL [43] as done in mono PL-SLAM [35] brings no significant improvement. In this work, we then utilize the BA with 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 [35] replicated from EPnPL, which means ours is naturally more efficient and avoids the shifting ambiguity of the line.

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. In Table I and II 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 observed in the 3rd column (**ours**: point + line).

The implicit constraint from 3D planar structures. Interestingly, in Table II, one can observe that our implicit constraint that minimizes the distance between the 3D point and 3D plane brings performance improvement on 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 on ICL-NUIM dataset, 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 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

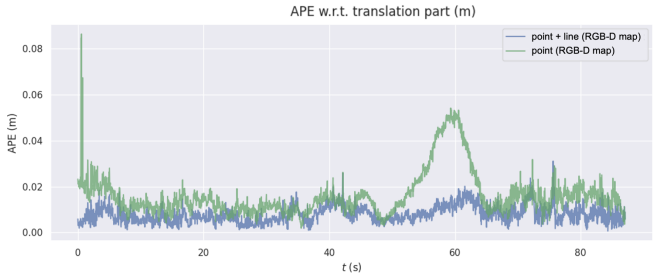


Fig. 5: **The evaluation of re-localization module.** We illustrate the absolute pose error (APE) [m] of all the relocalized images from sequence *freiburg3_long_office_household*, where x-axis indicates the relative timestamps.

problem with such unary edges 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 Fig. 4, we illustrate reconstructed point-line maps and line-plane maps selectively. Our map representation is designed as a lightweight sparse map, which intuitively shows the scene structure without heavy memory consumption. 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 add-on computation. 3D lines are naturally observable. It can illustrate most of the edge features of the scene.

In our framework, failure may happen to the planar mapping, when the instance planar segmentation is too noisy or even failed on unseen images. For that reason, we utilize graph-structure optimization in RANSAC to fit planes. However, it is not avoidable that there are many user-defined thresholds that are difficult to fine-tune. We solve this partially by employing the adaptive parameter setting strategy. Moreover, the 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 also results in mapping failure in no-texture scenes, as shown in Table IV.

C. Re-localization Module

Re-localization module estimates the camera pose without using any prior information other than the map. This is useful

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]**.

when the previous camera location cannot be used, such as when the tracking is lost. It is recently addressed as the map-based visual localization problem [12], [49], [50] or global localization [34], [37], just to name a few.

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, the blue line indicates using points and lines for pose optimization (based on a pre-built RGB-D point-line map), which shows obvious smaller errors compared to the one (green line) using points only (based on a pre-built RGB-D point cloud map).

IV. Conclusion

We presented a sparse visual SLAM that utilizes points and line segments for robust camera localization, while planes are an add-on to the map to intuitively illustrate the structure of the scene. Comprehensive evaluations are given, with qualitative results of various reconstructed maps, which verifies the solid implementation of ours. As the system was designed based on monocular assumption, future work would be generalizing this full SLAM system to more challenging scenarios, such as in no-texture scenes. Utilizing other high-level features such as 3D objects is also of high interest.

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.

V. Supplementary Materials

A. Hidden Parameters of LSD Extractor

Name of the Parameter	Value
opts.refine	1
opts.scale	0.5
opts.sigma_scale	0.6
opts.quant	2.0
opts.ang_th	22.5
opts.log_eps	1.0
opts.density_th	0.6
opts.n_bins	1024
opts.min_length	0.125 * min(img.cols, img.rows)

TABLE V: **The hidden parameters used for extracting 2D line segments using LSD**, as discussed in PL-VINS [10], which can be 3 times faster than original implementation in OpenCV. The meaning of the parameters please refer to the OpenCV documentations of LSD.

B. Conversion between Orthonormal Representation and Plücker Coordinates

As an infinite line representation, the Plücker coordinates $\mathbf{L}_w = (\mathbf{m}_w^\top, \mathbf{d}_w^\top)^\top$ can be calculated by intersecting two 3D planes as discussed in two-view triangulation, or constructed by two 3D endpoints $\mathbf{X_1} = (X_1, Y_1, Z_1, 1)^\top$ and $\mathbf{X_2} = (X_2, Y_2, Z_2, 1)^\top$:

$$\mathbf{L}_w = \begin{bmatrix} \mathbf{m}_w \\ \mathbf{d}_w \end{bmatrix} = \begin{bmatrix} Y_2Z_1 - Y_1Z_2 \\ Z_2X_1 - Z_1X_2 \\ X_2Y_1 - X_1Y_2 \\ X_1 - X_2 \\ Y_1 - Y_2 \\ Y_1 - Y_2 \end{bmatrix} \tag{11}$$

The orthonormal representation $(\mathbf{U}, \mathbf{W}) \in SO(3) \times SO(2)$ of a 3D line can be computed using the QR decomposition given the the Plücker coordinates [10]:

$$[\mathbf{m}_w \mid \mathbf{d}_w] = \mathbf{U} \begin{bmatrix} \omega_1 & 0 \\ 0 & \omega_2 \\ 0 & 0 \end{bmatrix}, with: \mathbf{W} = \begin{bmatrix} \omega_1 & \omega_2 \\ -\omega_2 & \omega_1 \end{bmatrix} \tag{12}$$

where $\mathbf{U}$ and $\mathbf{W}$ denote a three and a two dimensional rotation matrix, respectively. Let $\mathbf{R}(\boldsymbol{\theta}) = \mathbf{U}$ and $\mathbf{R}(\theta) = \mathbf{W}$ be the corresponding rotation transformations, we have:

$$\begin{aligned} \mathbf{R}(\boldsymbol{\theta}) &= [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = \left[\frac{\mathbf{m}_w}{||\mathbf{m}_w||}, \frac{\mathbf{d}_w}{||\mathbf{d}_w||}, \frac{\mathbf{m}_w \times \mathbf{d}_w}{||\mathbf{m}_w \times \mathbf{d}_w||} \right] \\ \mathbf{R}(\theta) &= \begin{bmatrix} \omega_1 & \omega_2 \\ -\omega_2 & \omega_1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \\ &= \frac{1}{\sqrt{(||\mathbf{m}_w||^2 + ||\mathbf{d}_w||^2)}} \begin{bmatrix} ||\mathbf{m}_w|| & -||\mathbf{d}_w|| \\ ||\mathbf{d}_w|| & ||\mathbf{m}_w|| \end{bmatrix} \end{aligned} \tag{13}$$

where $\boldsymbol{\theta}$ and θ denote a 3-vector and a scalar, respectively.

Within the iterative optimization in the BA, $\mathbf{U}$ and $\mathbf{W}$ can be updated as $\mathbf{U} \leftarrow \mathbf{U}\mathbf{R}(\boldsymbol{\theta})$ (notice that $\boldsymbol{\theta} \in \mathbb{R}^3$) and $\mathbf{W} \leftarrow \mathbf{W}\mathbf{R}(\theta)$ (notice that $\theta \in \mathbb{R}$). Therefore the orthonormal representation has the minimal 4 parameters as $\left[\boldsymbol{\theta}^\top, \theta\right] \in \mathbb{R}^4$.

Given an orthonormal representation $(\mathbf{U}, \mathbf{W})$, we can recover its Plücker coordinates by:

$$\mathbf{L}_w = \left[\omega_1 \mathbf{u}_1^\top, \omega_2 \mathbf{u}_2^\top\right], \tag{14}$$

where ω_1, ω_2, $\mathbf{u}_1$, and $\mathbf{u}_2$ can be extracted from Eq. (13), as $\mathbf{u}_i$ the i-th column of $\mathbf{U}$.

C. The Analytical Jacobians of 3D Line

The complete Jacobians of reprojection error of the line [25] regarding to the **orthonormal representations** (see last Sec. V-B) and **camera poses** $\mathfrak{se}(3)$ are:

$$\mathbf{J}_{\boldsymbol{\theta}} = \frac{\partial \mathbf{e}_l}{\partial \boldsymbol{\delta}_{\boldsymbol{\theta}}} = \frac{\partial \mathbf{e}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_c} \frac{\partial \mathbf{L}_c}{\partial \mathbf{L}_w} \frac{\partial \mathbf{L}_w}{\partial \boldsymbol{\delta}_{\boldsymbol{\theta}}} \tag{15}$$

$$\mathbf{J}_{\xi} = \frac{\partial \mathbf{e}_l}{\partial \boldsymbol{\delta}_{\xi}} = \frac{\partial \mathbf{e}_l}{\partial \mathbf{l}} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_c} \frac{\partial \mathbf{L}_c}{\partial \boldsymbol{\delta}_{\xi}} \tag{16}$$

where the partial derivative of the line reprojection error w.r.t the reprojected line $\mathbf{l}$ is:

$$\frac{\partial \mathbf{e}_l}{\partial \mathbf{l}} = \frac{1}{\sqrt{l_1^2 + l_2^2}} \begin{bmatrix} x_s - \frac{l_1 \mathbf{x}_s \mathbf{l}}{\sqrt{l_1^2+l_2^2}} & y_s - \frac{l_2 \mathbf{x}_s \mathbf{l}}{\sqrt{l_1^2+l_2^2}} & 1 \\ x_e - \frac{l_1 \mathbf{x}_e \mathbf{l}}{\sqrt{l_1^2+l_2^2}} & y_e - \frac{l_2 \mathbf{x}_e \mathbf{l}}{\sqrt{l_1^2+l_2^2}} & 1 \end{bmatrix}_{2\times 3} \tag{17}$$

The partial derivatives of the reprojected line $\mathbf{l}$ w.r.t $\mathbf{L}_c$, and $\mathbf{L}_c$ w.r.t $\mathbf{L}_w$:

$$\begin{aligned} \frac{\partial \mathbf{l}}{\partial \mathbf{L}_c} &= \frac{\partial \mathbf{K}_L \mathbf{m}_c}{\partial \mathbf{L}_c} = \begin{bmatrix} \mathbf{K}_L & \mathbf{0}_{3\times 3} \end{bmatrix}_{3\times 6} \\ \frac{\partial \mathbf{L}_c}{\partial \mathbf{L}_w} &= \frac{\partial \mathbf{T}_{cw} \mathbf{L}_w}{\partial \mathbf{L}_w} = \mathbf{T}_{cw} = \begin{bmatrix} \mathbf{R}_{cw} & [\mathbf{t}_{cw}]_\times \mathbf{R}_{cw} \\ \mathbf{0}_{3\times 3} & \mathbf{R}_{cw} \end{bmatrix}_{6\times 6} \end{aligned} \tag{18}$$

The derivative of $\mathbf{L}_w$ w.r.t the orthonormal representation is:

$$\frac{\partial \mathbf{L}_w}{\partial \boldsymbol{\delta}_{\boldsymbol{\theta}}} = \begin{bmatrix} \mathbf{0}_{3\times 1} & -\omega_1 \mathbf{u}_3 & \omega_1 \mathbf{u}_2 & -\omega_2 \mathbf{u}_1 \\ \omega_2 \mathbf{u}_3 & \mathbf{0}_{3\times 1} & -\omega_2 \mathbf{u}_1 & \omega_1 \mathbf{u}_2 \end{bmatrix}_{6\times 4} \tag{19}$$

The derivative of $\mathbf{L}_c$ w.r.t camera pose is:

$$\frac{\partial \mathbf{L}_c}{\partial \boldsymbol{\delta}_{\xi}} = \begin{bmatrix} -[\mathbf{R}\mathbf{m}]_\times - [[\mathbf{t}]_\times \mathbf{R}\mathbf{d}]_\times & -[\mathbf{R}\mathbf{d}]_\times \\ -[\mathbf{R}\mathbf{d}]_\times & \mathbf{0}_{3\times 3} \end{bmatrix}_{6\times 6} \tag{20}$$

D. Endpoints Trimming

Given an observed 2D line segment with endpoints $\{\mathbf{x}_s, \mathbf{x}_e\}$, and correspondingly a reprojected line $\mathbf{l}$, an perpendicular intersection plane is constructed via finding the closet point $\mathbf{x}_\perp$ of $\mathbf{x}_s$ (or $\mathbf{x}_e$) to the line $\mathbf{l}$ [25], [52]:

$$x_{\perp s} = -\left(y_s - \frac{l_2}{l_1} x_s + \frac{l_3}{l_2}\right) \frac{l_1 l_2}{l_1^2 + l_2^2} \tag{21}$$

$$y_{\perp s} = -\frac{l_1}{l_2} x_{\perp s} - \frac{l_3}{l_2} \tag{22}$$

Calculating a random point $\mathbf{x_{0s}}$, e.g., with $x_{0s} = 0$:

Monocular	**Ours** (full)	**Ours**	OpenVSLAM [42]
Config.	point + line + plane	point + line	point
MH_01_easy	-	0.043	0.045
MH_02_easy	-	0.035	0.037
MH_03_medium	-	0.038	0.039
MH_04_difficult	-	0.155	0.177
MH_05_difficult	-	0.088	0.075
Avg.	-	**0.072**	0.074
V1_01_easy	0.093	0.096	0.095
V1_02_medium	0.062	0.065	0.064
V1_03_difficult	0.070	0.069	0.069
V2_01_easy	0.058	0.059	0.063
V2_02_medium	0.057	0.057	0.064
V2_03_difficult	0.151	0.106	0.127
Avg.	0.082	**0.075**	0.080

TABLE VI: **Monocular SLAM** evaluated on dataset **EuRoC MAV** [4], presented are the **absolute trajectory errors (ATEs) RMSE [m]** (- stands result not available due to the failure of instance planar segmentation CNN on factory image sequences *MH_01 - 05*). Each result from ours was calculated as the average over 5 executions.

Stereo	**Ours** (full)	**Ours**	OpenVSLAM [42]
Config.	point + line + plane	point + line	point
MH_01_easy	-	0.046	0.050
MH_02_easy	-	0.056	0.058
MH_03_medium	-	0.048	0.053
MH_04_difficult	-	0.071	0.072
MH_05_difficult	-	0.071	0.064
Avg.	-	0.059	0.059
V1_01_easy	0.089	0.091	0.093
V1_02_medium	0.066	0.066	0.067
V1_03_difficult	0.064	0.065	0.073
V2_01_easy	0.060	0.061	0.061
V2_02_medium	0.060	0.061	0.064
V2_03_difficult	0.163	0.166	0.157
Avg.	**0.084**	0.085	0.086

TABLE VII: **Stereo SLAM** evaluated on dataset **EuRoC MAV** [4], presented are the **absolute trajectory errors (ATEs) RMSE [m]**.

$$x_{0s} = 0, y_{0s} = y_s - \frac{l_2}{l_1} x_s \tag{23}$$

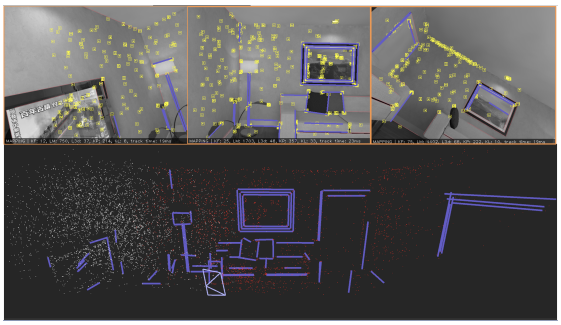
By doing that, we could construct 3D plane by:

$$\boldsymbol{\pi}_s = \mathbf{P}^\top \mathbf{l}_{\perp s} \tag{24}$$

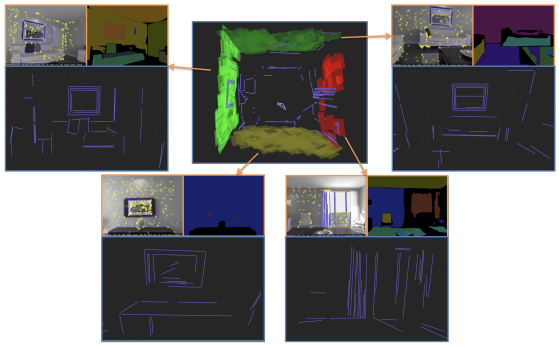
where $\mathbf{l}_{\perp s} = \mathbf{x}_{\perp s} \times \mathbf{x_{0s}}$. Given the Plücker coordinates of the 3D line, we are able to calculate the 3D starting point via intersecting the 3D plane and 3D line [15] via $\mathbf{X}_s = (X_s/\omega, Y_s/\omega, Z_s/\omega, 1)^\top = \mathbf{L}\boldsymbol{\pi}_s$, where:

$$\mathbf{L} = \begin{bmatrix} [\mathbf{m}]_\times & \mathbf{d} \\ -\mathbf{d}^\top & 0 \end{bmatrix} \tag{25}$$

The ending point of the 3D line is estimated by the same procedure described above for the starting point.



(a) **Point-line** map reconstructed from our **monocular SLAM**, of data sequence *living_room_traj0*.



(b) **Line-plane** map (middle) and **point-line** map reconstructed from our **monocular SLAM**, of data sequence *living_room_traj2*.

Fig. 6: **A qualitative illustration** of the map reconstructed from our **monocular SLAM**, on ICL-NUIM dataset [14].

E. More Quantitative and Qualitative Results

Dataset EuRoC MAV. As presented in Table VI and Table VII, we tested our monocular and stereo SLAM on dataset EuRoC MAV [4] which consists of 11 stereo sequences recorded with a MAV flying across three different environments: two indoor rooms and one industrial scenario, containing sequences that present different challenges depending on the speed of the drone, illumination, texture, etc. Note that results of exploiting planar reconstruction are not available on factory image sequences *MH_01 - 05*, due to the failure of instance planar segmentation CNN.

The exploitation of line segments barely increases the accuracy on dataset EuRoC MAV, similar results were also mentioned in stereo PL-SLAM [13]. Thus, stereo PL-SLAM only reports the relative pose errors of the keyframes, which are not compared in Table VII using ATEs. Still, our SLAM system shows slightly superior performance for most of the sequences in comparison to the point-only approach, especially the reconstructed map is more intuitive.

More qualitative illustration. More examples from our monocular SLAM on ICL-NUIM dataset see Fig. 6. Another example of our RGB-D SLAM is given in Fig. 7. As mentioned in the main paper, we also provide a comparison between ours and PL-VINS [10] in Fig. 8, qualitatively showing that both our monocular and stereo SLAM provides highly accurate point and line maps. Another example with planar reconstruction is given in Fig. 9.

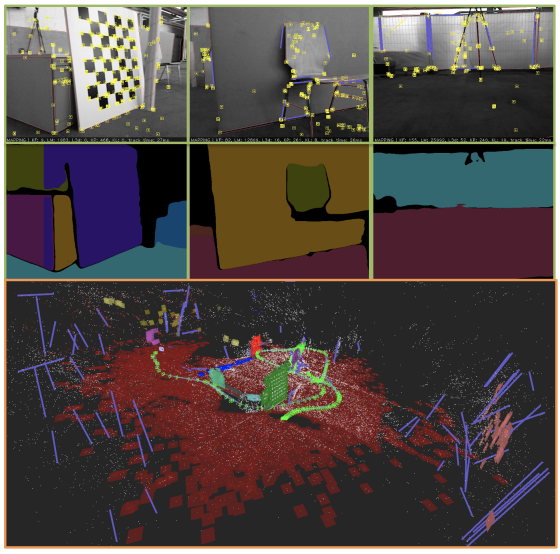
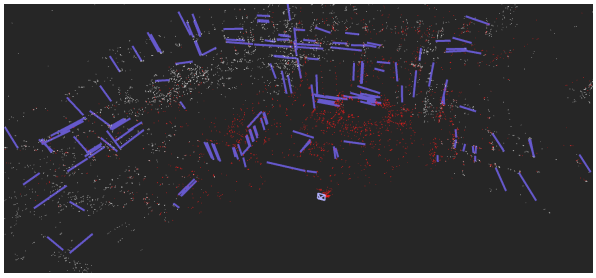
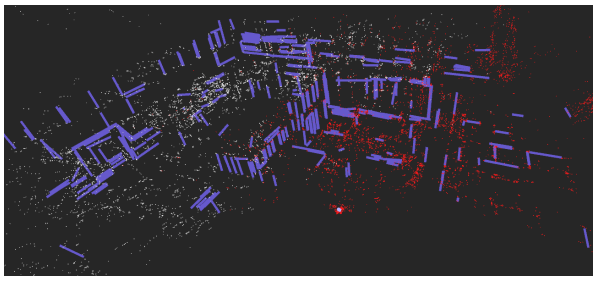


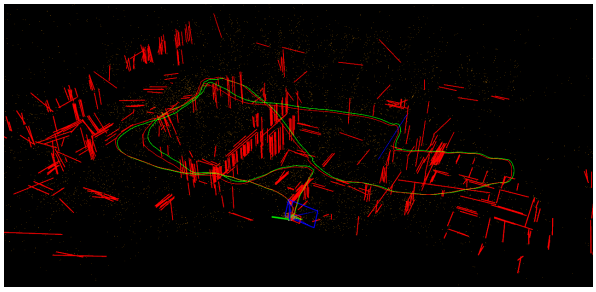
Fig. 7: **A qualitative illustration** of the **point-line-plane** map reconstructed from our **RGB-D SLAM**, on TUM RGB-D dataset [41], of data sequence *fr2_pioneer_slam*.



(a) Point-line map reconstructed from our **monocular SLAM**.

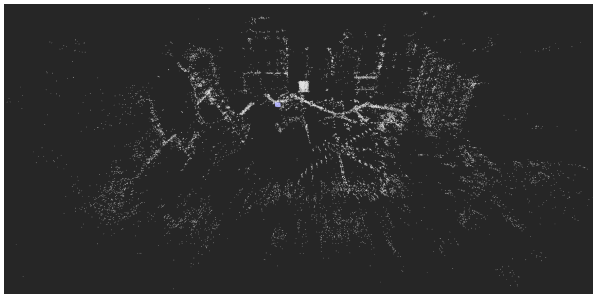


(b) Point-line map reconstructed from our **stereo SLAM**.

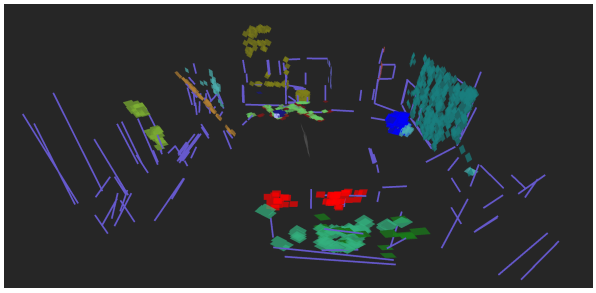


(c) Point-line map reconstructed from **PL-VINS**.

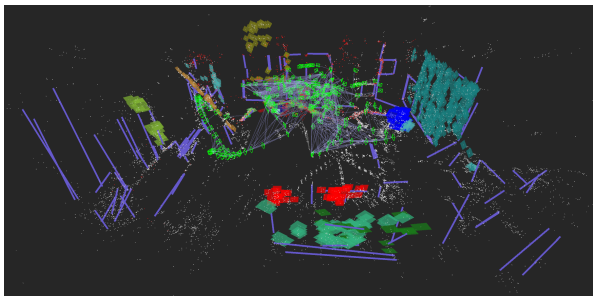
Fig. 8: **A qualitative comparison between our stereo SLAM and PL-VINS** [10], on *MH_04_difficult* of EuRoC dataset [4].



(a) Point cloud map reconstructed from our **stereo SLAM**.



(b) Line-plane map reconstructed from our **stereo SLAM**.



(c) Full visualization of map reconstructed from our **stereo SLAM**.

Fig. 9: **A qualitative illustration** of our **stereo SLAM**, on *V1_03_difficult* of EuRoC dataset [4].