Do Planar Constraints Improve Camera Pose Estimation in Monocular SLAM?

Charlotte Arndt
Robert Bosch GmbH &
University of Zaragoza
charlotte.arndt2@de.bosch.com

Reza Sabzevari
Center for Artificial Intelligence
Bosch Research, Germany
reza.sabzevari@de.bosch.com

Javier Civera
University of Zaragoza
Zaragoza, Spain
jcivera@unizar.es

Abstract

Geometric structures such as lines and planes are relevant in SLAM, as they improve the map interpretability and usability for downstream tasks. Planar landmarks add structural constraints to the map optimization, which could improve the accuracy of camera pose estimates. However, does the latter really happen in practice? In this paper, we thoroughly evaluate the effect of adding planar constraints in monocular SLAM, both in simulated and real scenes. Our experiments show that, in practical use cases, the benefit of adding such planar constraint shows benefits for scene estimation but limited effect in the camera pose estimation.

1. Introduction

Simultaneous Localisation and Mapping (SLAM) comprises a wide array of methods for robotic navigation in unknown environments, which have experienced significant progress along many relevant directions in the last decades [3]. Specifically, in visual SLAM, there have been substantial efforts to move from methods that only use point landmarks [19, 9] to higher level scene representations that can be formed by lines [24, 20, 10], planes [29, 1] or objects [4, 27, 14]. Such higher level representations make the resulting maps interpretable by humans and usable for high-level robot tasks like fetching objects.

The usual strategy for adding such high-level elements is to gather all of them in a single state vector and to optimize them jointly. This has been shown to improve the estimation of the SLAM state in several cases, e.g., [22, 2]. In monocular SLAM, the addition of line features to the state vector has also been shown to improve the accuracy of the camera trajectory [20, 10]. However, the effect in the state of adding planar features has not been sufficiently studied.

In this paper, we focus on the specific case of adding planes to the joint optimization in *monocular* SLAM. We build on our previous work [1], from which we use the state definition and measurement models. The main contribution of this paper is an analysis of the impact of adding planar constraints on the state accuracy. Such an analysis is done both in simulation and in real scenes. Our experiments suggest that the effect of adding planar constraints on camera pose estimation is limited. This is even more evident in practical situations. Specifically, we observed only slight improvements in the estimations of camera poses in several simulation setups, in which errors associated to plane extraction and matching are non-existent and we can synthetically assign noise levels and points to the corresponding planes. These and other effects present in real scenes might have in fact even a larger influence on the estimation errors than the benefit planes add. Generally, the joint optimization of planes with the rest of the state does not bring a significant advantage. This is in line with our experiments in real scenes.

The remainder of the paper is structured as follows: In Section 2 the related work on SLAM systems using planes is discussed. The formulation for joint optimization of points, planes and camera poses used in this analysis is introduced in Section 3. Section 4 details our experimental setup as well as the results. We summarize our conclusions in Section 5.

2. Related work

2.1. RGB-D Plane SLAM

With RGB-D cameras, it is possible to measure planes directly. Based on this, Kaess [15] introduces a minimal representation of planes for SLAM suitable for optimization with incremental solvers. Similar to quaternions, a plane in homogeneous representation can be normalised to lie on the unit sphere $\mathbb{S}^3$ and be updated in the tangent space. This resulted in a mapping system using only planes as landmarks.

Hosseinzadeh et al. [13] included this minimal formulation in their RGB-D SLAM system which uses points, planes and objects (represented as quaternions) in the map. The distance of points to their corresponding plane is minimized. Additionally, they include constraints between planes based on the Manhattan assumption and a tangen-

tial constraint between objects and planes. They are able to show substantial improvement compared to RGB-D ORB-SLAM2 [19].

Zhou et al. [30] focus on the least-squares problem a SLAM system with planar landmarks needs to solve. They point out the challenges for a plane-to-plane cost and propose an efficient algorithm for a distance-based point-to-plane factor.

2.2. Monocular Plane SLAM

In contrast to 3D data, a single monocular image does not generally contain the geometric information to observe the parameters of a plane. However, several monocular SLAM systems incorporating multi-view planes have been proposed.

Wu and Beltrame [26] propose a visual odometry system using planes, extracted by a deep network, and only photometric error terms. It improves over the baseline without planes. Rosinol et al. [21] introduce a monocular visual-inertial 3D mesh generator, which makes use of planarity constraints among mesh triangles. However, they only consider horizontal and vertical planes. Yang et al. [29] estimate wall and floor planes from wall-floor boundaries, which they detect using a CNN and edge detection. The plane landmarks improve the robustness of their SLAM system in scenes with little texture. While this is an important use case for adding planes, the authors do not evaluate their system in general scenes. As a follow-up, [28] adds objects and modifies the plane detection. Pixelwise semantic segmentation is used to find initial wall-ground boundaries and the detections are refined using a CRF. Wall-ground boundary lines of planes are used to measure planes in 2D images and therefore only wall planes (no ground) are used. While the system can improve over the baseline in many evaluated sequences, the normal direction of the planes is fixed to be parallel to the Manhatten world axis directions.

Hosseinzadeh et al. propose a monocular SLAM system using points, planes and objects [14]. The planes are extracted by a neural network which makes it possible to generate plane measurements from monocular images directly. The distance between points and their associated plane is minimized. Objects are represented as quadrics. The system makes use of the Manhattan world assumption to add constraints which reward parallel or orthogonal planes. They can show a slight improvement of trajectory accuracy using planes but report only one run per sequence. In [1], a monocular SLAM system is presented, that in contrast to most other works enforces points associated with planes to lie exactly inside this plane. This system is capable of representing general planes.

Some methods decouple estimation of planes and map state. Concha and Civera [6, 7] use triangulated planes to complete sparse and semi-dense monocular maps, but do not optimize the planes jointly with the map states. Shu et al. [23] use a CNN to detect planes in a multi-plane SLAM system decoupled from local bundle adjustment. The system can be used in monocular and RGB-D settings. Plane detections are further refined using a graph-cut RANSAC approach which also enforces spatial consistency for both inliers and outliers. In [16] Li et al. decouple rotation and translation estimation to achieve low drift in a monocular SLAM system. Lines and planar areas are used during the rotation estimation leveraging the Manhattan world assumption. Translation estimation is based on points and lines.

2.3. Plane SLAM with Manhattan World

Many works using planes for localization and mapping are based on the Manhattan World assumption [8]. [21, 29, 28], among others, impose this assumption in the plane discovery phase by allowing planes only in Manhattan directions. Other works such as [14] add the Manhattan assumptions explicitly as additional constraints to the optimization problem. Using such constrains makes sense in man-made environments and improves the quality of SLAM systems [5]. However, in this work we are solely looking into the benefits of adding planar constraints to the optimisation problem of the SLAM back-end. In our experiments, we have isolated the problem to study the benefits of using planar constraints and investigate in which circumstances they will improve the mapping and localization accuracy. We choose to use the pipeline proposed in [1] as it is free from assumptions like Manhattan World that may affect our study and considers general planes oriented in arbitrary directions.

We found the experiments in the literature to be limited for the following reasons: They report the influence of planar constraints in SLAM accuracy only under certain assumptions (e.g. Manhattan world), the reported influence of planar constraints in SLAM is not independent of the quality of plane discovery method, and a fair comparison of such methods is not possible, as different assumptions, datasets, and baselines are used. To study the impact of general planar landmarks, in this work we employ both synthetic and real sequences, set the state of the art point-feature based SLAM system [19] as the baseline and report to what extent a SLAM system can benefit from integrating planar landmarks.

3. SLAM with Points and Planes

This section introduces the formulation for optimizing points and planes in monocular SLAM system used for during the experiments. We follow our formulation previously presented in [1] and we use it with minimal modifications.

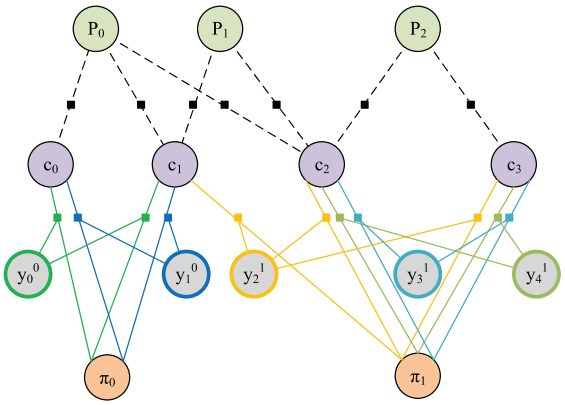


Figure 1: Factor graph of the optimization, figure taken from [1]. Vertices for 3D points are in green, camera poses in purple, planes in orange and in-plane points in grey with colored outline. Ternary factors connect camera poses, in-plane points and planes. Factors and edges leading to them are colored according to the in-plane point they are connected with. Figure best viewed in color.

3.1. Mapping and Tracking States

$$\boldsymbol{x}_M = \Big(\boldsymbol{c}_1^\top, \ldots, \boldsymbol{c}_i^\top, \ldots, \boldsymbol{P}_1^\top, \ldots, \boldsymbol{P}_j^\top, \ldots, \boldsymbol{\pi}_1^\top, \ldots, \boldsymbol{\pi}_k^\top, \ldots, \boldsymbol{y}_1^{1\top}, \ldots, \boldsymbol{y}_l^{k\top}, \ldots\Big)^\top \quad (1)$$

where $\mathcal{C} = \{\boldsymbol{c}_i \mid i = 1, \ldots, m\,; \boldsymbol{c}_i \in SE(3)\}$ represents the poses of a set of keyframes, $\mathcal{P} = \{\boldsymbol{P}_j \mid j = 1, \ldots, n\,; \boldsymbol{P}_i \in \mathbb{R}^3\}$ are regular 3D points that do not belong to planes, $\mathcal{K} = \{\boldsymbol{\pi}_k \mid k = 1, \ldots, p\,; \boldsymbol{\pi}_k \in \mathbb{P}^3\}$ are a set of 3D planes in the scene, and $\mathcal{Y} = \{\boldsymbol{y}_l^k \mid k = 1, \ldots, p\,; l = 1, \ldots, n_k\,; \boldsymbol{y}_l^k \in \mathbb{R}^2\}$ are in-plane points, expressed in the local reference system of the 3D plane. The tracking state $\boldsymbol{x}_t \in SE(3)$ contains the camera pose in the current time step t.

3.2. Plane Representation

A plane in homogeneous representation is given as $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)^\top \in \mathbb{P}^3$. Its normal vector $\mathbf{n} = (n_x, n_y, n_z)^\top$ and the distance to the origin d are as follows:

$$\boldsymbol{n} = \frac{(\pi_1, \pi_2, \pi_3)^\top}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}} \qquad d = \frac{-\pi_4}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}} \quad (2)$$

In order to optimize the parameters of a plane, it is best to use a minimal representation. This can be achieved by optimizing in the tangent space of S^3 as proposed in [15].

Normalizing the homogeneous plane coordinates gives us $\boldsymbol{\pi}' = \boldsymbol{\pi}/\|\boldsymbol{\pi}\| \in S^3$. A quaternion $\boldsymbol{q}$ can also be normalised to a unit quaternion $\boldsymbol{q}' \in S^3 = \{\boldsymbol{q} \in \mathbb{R}^4 : \|\boldsymbol{q}\| = 1\}$ which is frequently used to represent and optimize rotations. Both representations cover the same space and therefore, they can be optimized in the same tangent space of S^3 using the exponential and log mapping derived by Grassia in [11]:

$$\boldsymbol{q}^{(s+1)} = \exp(\boldsymbol{\omega})\boldsymbol{q}^{(s)}, \;\; \exp(\boldsymbol{\omega}) = \begin{pmatrix} \frac{1}{2} sinc(\frac{1}{2}\|\boldsymbol{\omega}\|)\boldsymbol{\omega} \\ \cos(\frac{1}{2}\|\boldsymbol{\omega}\|) \end{pmatrix} \quad (3)$$

where s is an iteration step of the optimization and $\boldsymbol{\omega} \in \mathbb{R}^3$ represents an incremental update of the plane.

3.3. In-plane point representation

In-plane points $\boldsymbol{y}_l^k \in \mathbb{R}^2$ are represented in the local coordinate frame of the plane. Their position relative to the plane origin $\boldsymbol{O}_k^w$ is represented as a linear combination of two arbitrary orthogonal in-plane vectors, v_0 and v_1. For convenience, the origin of plane $\boldsymbol{\pi}_k$ is defined as $\boldsymbol{O}_k^w = -d\boldsymbol{n}^\top$.

As $v_0 \cdot n = 0$ and we define it of unit length $\|v_0\| = 1$, arbitrarily setting the y-component to zero provides the necessary constraints to obtain it. The original formulation for v_0 in [1], however, had undesired properties during incremental updates of the plane parameters. Specifically, the sign for x-component of v_0 depended on the sign of the x-component of the plane normal n. As a result, small updates in the plane parameters inducing a sign change might result in an abrupt change of the vectors v_0 and v_1. This sign change can be prevented by a more stable formulation for v_0:

$$\boldsymbol{v}_0 = \begin{pmatrix} \frac{-n_z}{\sqrt{{n_z}^2 + {n_x}^2}} & 0 & \frac{n_x}{\sqrt{{n_z}^2 + {n_x}^2}} \end{pmatrix}^\top \quad (4)$$

As v_1 is perpendicular to both v_0 and $\boldsymbol{n}$, it can be determined as $\boldsymbol{v}_1 = \boldsymbol{v}_0 \times \boldsymbol{n}$. Finally, we can convert a 3D point in the world coordinate system $\boldsymbol{P}_j \in \mathbb{R}^3$ into the in-plane point $\boldsymbol{y}_l^k$ in the reference plane of plane $\boldsymbol{\pi}_k$ by projecting it onto the plane along the normal direction.

$$\boldsymbol{y}_l^k = \begin{bmatrix} (\boldsymbol{P}_j - \boldsymbol{O}_k^w) \cdot \boldsymbol{v}_0 \\ (\boldsymbol{P}_j - \boldsymbol{O}_k^w) \cdot \boldsymbol{v}_1 \end{bmatrix} \quad (5)$$

3.4. Bundle Adjustment using Point and Plane Constraints

Our measurements $\mathcal{Z} = \{\boldsymbol{z}_{1,1}, \ldots, \boldsymbol{z}_{i,j}, \ldots, \boldsymbol{z}_{i_k,j_k,k}, \ldots\}$ are the image observations of regular points ($\boldsymbol{z}_{i,j}$) and in-plane points ($\boldsymbol{z}_{i_k,j_k,k}$). The cost to minimize is the sum of a function of the reprojection errors of all points in all images.

$$C = \sum_{i,j} \rho_h(\boldsymbol{e}_{i,j}^T \boldsymbol{\Omega}_{i,j}^{-1} \boldsymbol{e}_{i,j}) + \sum_{i,k,l} \rho_h(\boldsymbol{e}_{i,k,l}^T \boldsymbol{\Omega}_{i,k,l}^{-1} \boldsymbol{e}_{i,k,l}) \quad (6)$$

We use the Huber cost function ρ_h and $\mathbf{\Omega}_{i,j}$ stands for the covariance matrix for the keypoint. The projection equation is similar for 3D points and in-plane points. For 3D points, the keypoint $\boldsymbol{z}_{i,j}$ in the image of camera $\boldsymbol{c}_i$ is associated with point $\boldsymbol{P}_j$ and compared to its reprojected position.

$$\boldsymbol{e}_{i,j} = \boldsymbol{z}_{i,j} - proj(\boldsymbol{R}_i \boldsymbol{P}_j + \boldsymbol{t}_i) \tag{7}$$

where $\boldsymbol{R}_i$ stands for the rotation matrix that transforms points in the world frame to the camera frame, and $\boldsymbol{t}_i$ for the translation vector from the optical center of the camera to the world frame in the local camera frame. $proj(\cdot) : \mathbb{R}^3 \rightarrow \Omega$ projects a point in the camera frame to the image domain Ω.

An in-plane point $\boldsymbol{y}_l^k$ in a plane π_k can be converted to the world coordinate system $\boldsymbol{Y}_{k,l}^w$ at any time using

$$\boldsymbol{Y}_{k,l}^w = \begin{bmatrix} \boldsymbol{v}_0 & \boldsymbol{v}_1 \end{bmatrix} \cdot \boldsymbol{y}_l^k + \boldsymbol{O}_k^w \tag{8}$$

Afterwards, the point can be projected into the camera frame to calculate the error to the associated keypoint.

$$\boldsymbol{e}_{i,k,l} = \boldsymbol{z}_{i,k,l} - proj(\boldsymbol{R}_i \boldsymbol{Y}_{k,l}^w(\boldsymbol{\pi}_k, \boldsymbol{y}_l^k) + \boldsymbol{t}_i) \tag{9}$$

Fig. 1 shows the resulting factor graph visualising the optimization described by the equations above. All points in the same plane are indirectly connected with each other via the shared plane parameters.

4. Experimental Results

When investigating planes as landmarks in visual SLAM, the high complexity of the pipeline makes it difficult to isolate and properly analyze certain effects. Additionally, when working on real data, many recordings are required to cover different scenes and trajectories and 3D scene ground truth is hard to obtain. While this can be easily achieved using data from simulation, there is usually a gap with real data. Therefore, we conduct two types of experiments.

In the first set of experiments, bundle adjustment with planar constraints is examined in isolation using simulated configurations, thus, making it possible to directly control spatial configuration and number of planes, trajectory shape, point density and more. We omit the problem of plane extraction here and focus on the accuracy improvement due to the new type of constraints. These experiments are presented in section 4.1. We complement this simulations with a second set of experiments using the full pipeline on real sequences. They are presented in section 4.2 and focus on the potential benefits of planes in real applications. Real sequences are taken from the TUM RGB-D dataset [25]. Our implementation of the SLAM pipeline and optimization is based on [19, 1]. For trajectory alignment and computation of the Absolute Pose Error (APE), we use EVO [12].

4.1. Simulation Results

The simulation experiments focus on investigating the effects of planar constraints on the estimates accuracy after full bundle adjustment. We use the APE as a measure of trajectory accuracy and mean 3D point error as a measure of scene accuracy. These experiments do not cover plane discovery, the effects of outliers or incorrect association. Starting with the simplest case, we look into the effect of a single plane with different trajectories. This setup is also close to the experiments on real sequences with one dominant plane reported in [1]. Additionally, we conduct experiments with different numbers of planes and scene points to investigate the benefits of planar constraints depending on these environmental factors.

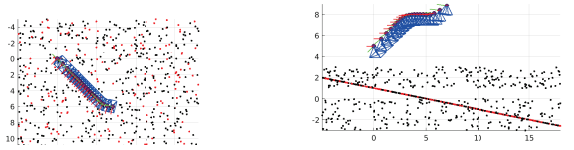
Experimental setup. We define a number of different camera trajectories with increasing complexity: a simple spline, a circular trajectory, and short and long curvy trajectories imitating the motion of a handheld camera. Top and side view of the scene are shown in the first row of Fig. 2 and 3. The camera poses are sampled from these trajectories and the 3D points are either inside a given 3D volume or lying on a plane. Data association is given. Points sampled from a plane might be associated with that plane or not, depending on the experiment. We compare levels of 0%, 50% and 100% of in-plane points. We apply zero-mean Gaussian noise to all point observations, with varying standard deviations (0.5, 1.0 or 1.5 pixel). This information is also passed to the optimization, replacing the noise parameters usually corresponding to different levels of the feature pyramid. The optimization seeds for the points positions and camera translations are perturbed with respect to the ground truth.

Single-plane experiments. We first evaluate four different trajectories in a scene with 3D points and one plane. We show the distribution of errors over 15 runs. Boxplots of the errors are shown in the bottom row of Fig. 2 and 3. The mean values are also reported in Table 1. In the table, we also report the relative change between 0% and 100% in-plane points for APE and 3D point errors. For 3D point errors, we report the mean taking into account all points (both in-plane and regular ones), only regular points or only in-plane points respectively.

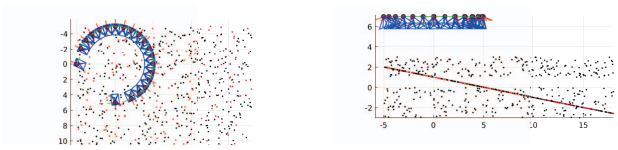
In single-plane experiments, the changes in mean APE for different ratios of in-plane points are minimal. Absolute improvements are in sub-millimeter range, the largest relative improvement below 4%. In comparison, the variation between runs due to noise visible in the boxplots is much larger. In contrast, the mean 3D point error decreases significantly for 50% and 100% of in-plane points compared to having no plane. The average improvement reaches some

Table 1: RMSE of APE and mean 3D point errors for different rates of in-plane points. Mean taken over 15 runs. Lower is better. Additional column with relative change from baseline (*) to 100% in-plane points (†).

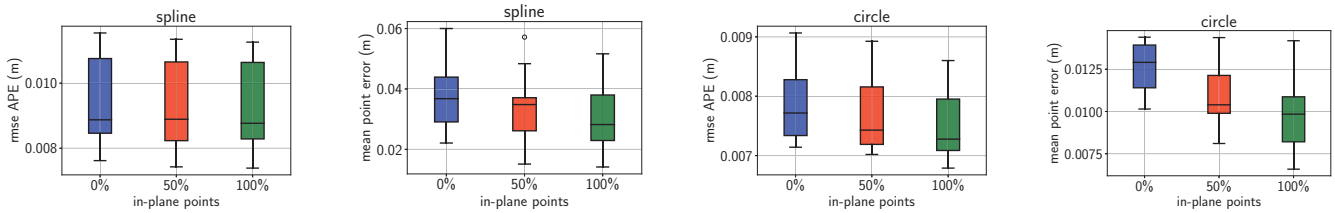
	rmse APE (cm)				mean 3D point error (cm)							
sequence	0% (*)	50%	100% (†)	rel. change (*,†)	0% all (*)	50% all	50% regular	50% in-plane	100% all(†)	100% regular	100% in-plane	rel. change (*,†)
spline	0.94	0.93	0.93	-1.4%	3.79	3.38	3.64	2.8	2.99	3.45	2.65	-21.1%
circle	0.78	0.77	0.75	-3.6%	1.26	1.09	1.24	0.71	0.97	1.236	0.71	-23.0%
handheld	0.85	0.85	0.84	-0.9%	1.14	1.02	1.23	0.52	0.843	2.37	0.51	-26.6%
handheld long	0.79	0.79	0.79	-0.7%	1.97	1.49	2.0	0.45	1.26	2.32	0.39	-36.1%
handheld - 2 planes	0.68	0.67	0.65	-4.0%	4.42	2.87	3.54	1.75	1.93	2.88	1.59	-56.3%
handheld - 5 planes	2.87	1.73	1.57	-45.3%	8.54	6.23	7.02	4.92	4.63	5.42	4.33	-45.8%



(a) top and side view of simple spline trajectory

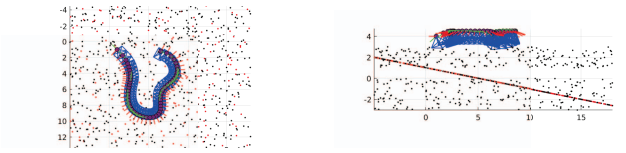


(b) top and side view of circle trajectory

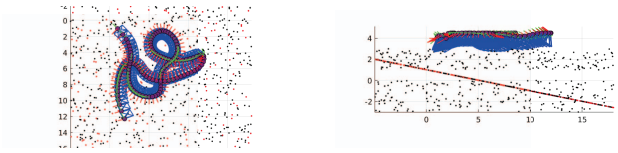


(c) Boxplots showing the distribution of APE and 3D point errors over 15 runs.

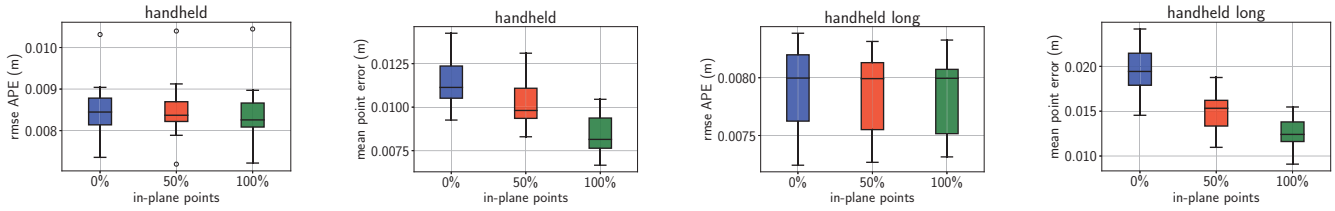
Figure 2: Scene and error boxplots for single-plane setups and short trajectories. Red scene points are optimized as in-plane points, here at 50% setting.



(a) top and side view of short handheld trajectory



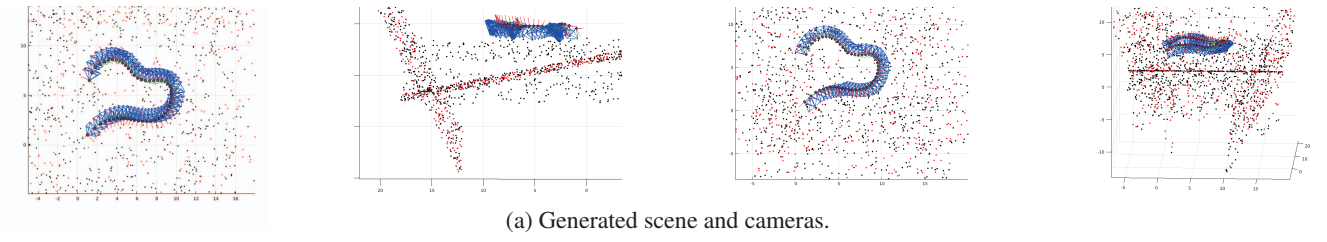
(b) top and side view of long handheld trajectory



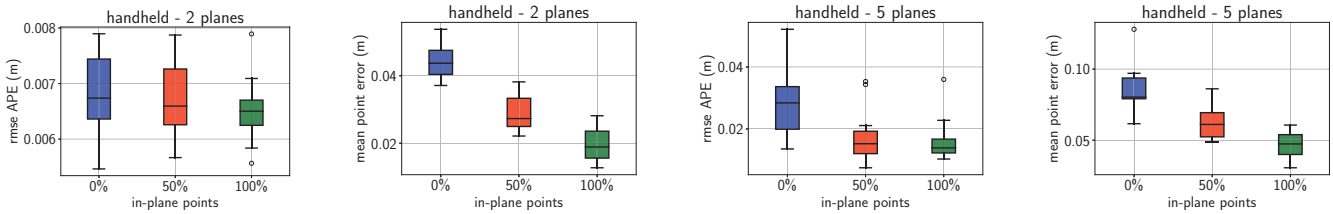
(c) Boxplots showing the distribution of APE and 3D point errors over 15 runs.

Figure 3: Scene and error boxplots for single-plane setups and large trajectories. Red scene points are optimized as in-plane points, here at 50% setting.

millimeters, which means relative improvements of more than 20%. When comparing the mean values of all, reg-



(a) Generated scene and cameras.



(b) Boxplots showing the distribution of errors over 15 runs.

Figure 4: Scene and error boxplots for multi-plane setups. Red scene points are optimized as in-plane points, here at 50% setting.

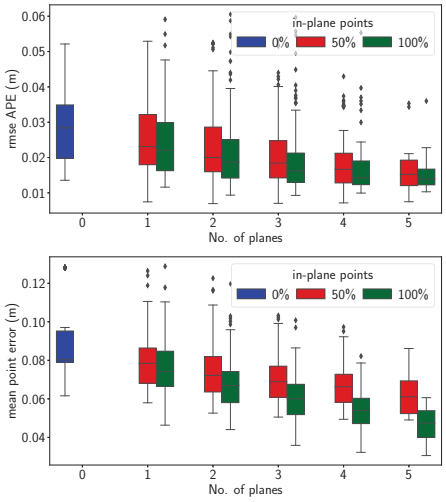


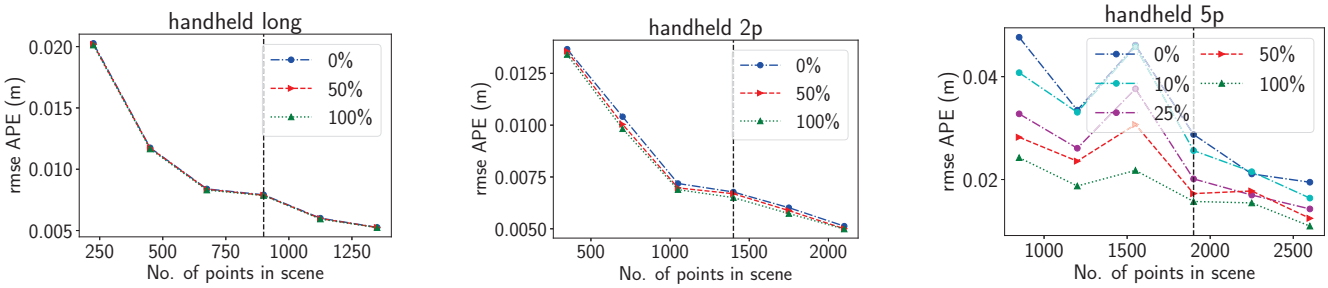
Figure 5: Error boxplots for the multi-plane case

ular and in-plane points, it becomes clear that this effect is mainly due to the reduced 3D point errors of in-plane points. Our results suggests that planar constraints have a large effect on the 3D scene accuracy, but a negligible one on the trajectory accuracy.

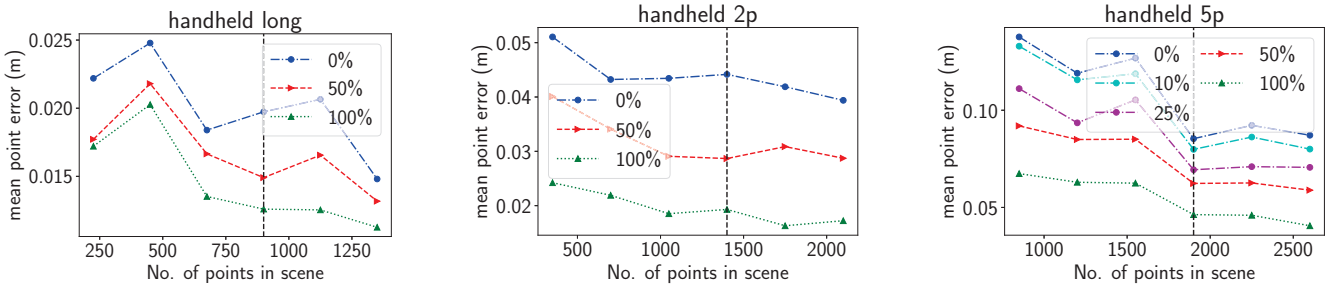
Multi-plane experiments. To explore the effect of planar constraints with multiple planes in the scene, we take the short handheld trajectory and add more planes. The camera orientation is changed for scenes with different numbers of planes to make sure the cameras also observe the new parts of the scene. Fig. 4 shows the results for 15 runs for two and five planes. The mean values are reported in Table 1. The trajectory errors decrease in less than a millimeter in the experiment with two planes, but more than a centimeter in the experiment with five planes. This corresponds to a reduction in error of 4% or 45% respectively when comparing having no in-plane points with the case of all possible in-plane points. The point error is reduced by 45% or more in both cases. So while the improvement in 3D point error is still the more dominant effect, in the 5-plane scene, the trajectory is also estimated more accurately using planar constraints.

We studied the 5-plane case in more detail by using planar constraints for different numbers of planes while keeping the same scene and the trajectory. We run experiments for all possible combinations of planes, that is, one experiment for the no-plane and the five-plane cases, then five experiments for the 1-plane and 4-plane cases, and ten experiments for all combinations of two planes and three planes. As before, we perform 15 runs per setting and report the results grouped by number of planes in Fig. 5. Our results show an improvement of the trajectory accuracy with an increasing number of planes and a high number of in-plane points. However, scenes such as these ones, with a high number of large planes and sufficient texture to extract local features, are unlikely to be found in practice. The point errors decrease significantly as we add planar constraints for more planes in the scene. There is also an advantage of 100% in-plane points compared to 50%, which grows with the number of planes.

Variations in the number of scene points. We conducted experiments with different numbers of 3D points to see if the effect of planes is larger in scenes with fewer constraints. We use all previous scene setups and trajectories but generate different numbers of 3D points while keeping the same ratio of points belonging to planes. The results for three of such trajectories are shown in Fig. 6. The three



(a) APE curves for different point densities.



(b) Point errors curves for different point densities.

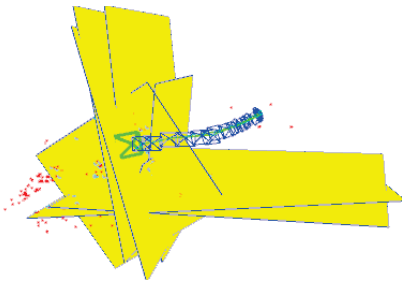
Figure 6: Errors depending on point densities. Dashed vertical line marks point density value shown in individual plots of each sequence in Fig. 3 and 4. Best viewed digitally.

simpler trajectories yield results qualitatively very similar to the handheld long trajectory, so we show this one as an example for all one-plane setups. For trajectory errors, the general trend is the same for all scenes. With a higher number of points in the scene the errors decrease, which is expected due to the higher number of constraints. However, there is no obvious effect on the differences between optimization with and without planar constraints. Having points generated more densely does not seem to have a direct effect on their effectiveness. The same is true for 3D point errors. The error generally decreases with more points in the scene and, similarly to previous experiments, the improvement for points is larger than for the trajectory. Again, there is no trend showing a stronger or weaker improvement when comparing optimization with and without planes depending on the number of points in the scene.

Summing up, our results indicate limited improvement of trajectory accuracy and clearly reduced estimation errors of the 3D points lying on the planes In realistic scenarios— where the number of planes is limited and the ratios of in-plane points is lower due to the lower number of tracked features— we expect improvement of camera pose accuracy to be further reduced compared to experiments in simulation.

4.2. Real-Image Results with a Full SLAM Pipeline

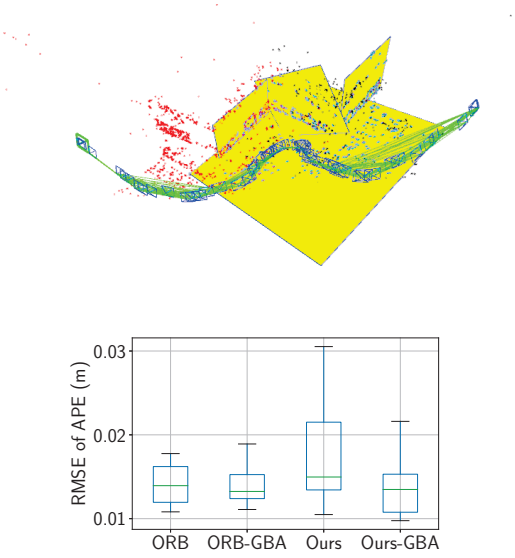
For experiments on real sequences, the formulation for planes and in-plane points was integrated into the monocular pipeline of the state-of-the-art ORB-SLAM2 [19] as proposed in [1]. For details on how different parts of the pipeline were modified, please refer to the original paper. We conduct experiments with multiple planes, and look into different methods for plane discovery.



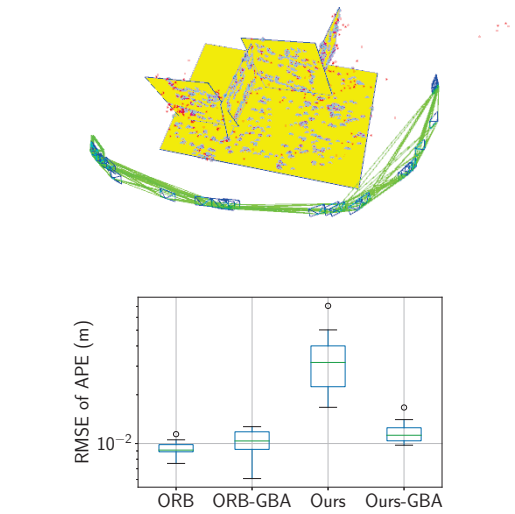
(a) Plane detections with RANSAC

(b) Plane detections with PlaneRCNN[17]

Figure 7: Plane detection methods failing on TUM dataset [25].



(a) str_txt_near[25]



(b) str_txt_far[25]

Figure 8: Maps and RMSE boxplots of experiments with multiple planes.

Specifically, for plane detection, we considered multiple methods. RANSAC or multimodel RANSAC can be used to find a plane hypothesis with a maximum number of inliers from a point cloud. However, the 3D point cloud constructed by the SLAM system does not sample structures homogeneously but depending on salient image regions where features can be matched. Therefore, RANSAC can easily lead to incorrect plane detections which do not correspond to real planar structures. Additionally, multimodel RANSAC needs careful parameter tuning, often sequence specific [18]. Neural networks such as PlaneRCNN [17] can also detect planar areas in 2D images. It is possible to estimate a plane hypothesis from all points falling inside the area of a mask. However, using them in images with significant domain change with respect to the training set leads to problems like over-segmentation. An example of failure cases for both RANSAC and PlaneRCNN segmentation is shown in Fig. 7.

For our experiments, we provide masks for individual planes derived from a reconstruction of the scene using RGB-D data and the ground truth trajectory instead of fine-tuning the network to our dataset. These was the most accurate plane discovery we could provide to make the experiment most competitive with the simulation and its perfect plane discovery. These masks come in a format provided by neural networks for plane segmentation like PlaneRCNN [17] but additionally provide a unique id for plane instances over the complete sequence. The resulting masks are used to segment sub-point-clouds corresponding to one plane each. We use single model RANSAC to find a plane hypothesis from this, combining the advances of both methods.

We report results on two sequences of the TUM dataset[25] with and without final global bundle adjustment. Maps and error boxplots are shown in Fig. 8. The reconstruction shows multiple visually correct planes inserted in the scene for both sequences. However, when comparing against the baseline, the trajectory errors and the variation between runs increase. By adding a final Global Bundle Adjustment (GBA), this effect is reduced. However, and in agreement with our simulation results, the trajectory errors remain similar to those of ORB-SLAM2 without planar constraints. Again, planar constraints offer a high-level representation of the scene but without a noticeable effect on the estimation of the camera poses.

5. Conclusions

In this paper we provided extensive evaluations of monocular mapping using planes on both synthetic data and real sequences. We find that planar constraints have a significant effect on the accuracy of the 3D structure estimation, specifically on the points lying on the plane. However, the effect in the camera trajectory accuracy is negligible. The same conclusion holds in our experiments in real scenes, where the benefits of planar constraints on the camera trajectory accuracy are again insignificant. Finally, we would like to remark that we do not want to diminish the value of planes as a high level representation that might be useful for certain applications. However, our findings suggest that the usual approach of jointly estimating the plane parameters along with the rest of state does not provide significant benefits for pose estimation in monocular setups and plane estimation could be decoupled with computation benefits.

References

[1] Charlotte Arndt, Reza Sabzevari, and Javier Civera. From points to planes - adding planar constraints to monocular SLAM factor graphs. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4917–4922. IEEE, 2020.

[2] Hriday Bavle, Jose Luis Sanchez-Lopez, Muhammad Shaheer, Javier Civera, and Holger Voos. Situational graphs for robot navigation in structured indoor environments. *IEEE Robotics and Automation Letters*, 2022.

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[4] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D Tardós, and JMM Montiel. Towards semantic slam using a monocular camera. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1277–1284. IEEE, 2011.

[5] Joan P Company-Corcoles, Emilio Garcia-Fidalgo, and Alberto Ortiz. Msc-vo: Exploiting manhattan and structural constraints for visual odometry. *IEEE Robotics and Automation Letters*, 7(2):2803–2810, 2022.

[6] Alejo Concha and Javier Civera. Using superpixels in monocular SLAM. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 365–372. IEEE, 2014.

[7] Alejo Concha and Javier Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693. IEEE, 2015.

[8] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 941–947. IEEE, 1999.

[9] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[10] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuniga-Noël, Davide Scaramuzza, and Javier 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.

[11] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[12] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[13] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Suenderhauf, and Ian Reid. Structure aware SLAM using quadrics and planes. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 410–426. Springer International Publishing, 2019.

[14] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-time monocular object-model aware sparse SLAM. In *2019 IEEE international conference on robotics and automation (ICRA)*, pages 7123–7129. IEEE, 2019.

[15] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[16] Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari. Structure-SLAM: Low-drift monocular SLAM in indoor environments. *IEEE Robotics and Automation Letters*, 5(4):6583–6590, 2020.

[17] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3d plane detection and reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4445–4454. IEEE, 2019.

[18] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3954–3961, 2014.

[19] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[20] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese 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.

[21] Antoni Rosinol, Torsten Sattler, Marc Pollefeys, and Luca Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8220–8226, 2019.

[22] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.

[23] Fangwen Shu, Yaxu Xie, Jason Rambach, Alain Pagani, and Didier 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, 2021.

[24] Joan Sola, Teresa Vidal-Calleja, Javier Civera, and José María Martínez Montiel. Impact of landmark parametrization on monocular ekf-slam with points and lines. *International Journal of Computer Vision*, 97(3):339–368, 2012.

[25] 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)*, pages 573–580, 2012.

[26] Fang Wu and Giovanni Beltrame. Direct sparse odometry with planes. *IEEE Robotics and Automation Letters*, 7(1):557–564, 2022.

[27] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-d object SLAM. *IEEE Transactions on Robotics*, pages 1–14, 2019.

[28] Shichao Yang and Sebastian Scherer. Monocular object and plane SLAM in structured environments. *IEEE Robotics and Automation Letters*, 4(4):3145–3152, 2019.

[29] Shichao Yang, Yu Song, Michael Kaess, and Sebastian 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.

[30] Lipu Zhou, Daniel Koppel, Hul Ju, Frank Steinbruecker, and Michael Kaess. An efficient planar bundle adjustment algorithm. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 136–145. IEEE, 2020.