3D Planar RGB-D SLAM System

Hakim ElChaoui ElGhor[1,2(✉)], David Roussel[1], Fakhreddine Ababsa[1], and El-Houssine Bouyakhf[2]

[1] IBISC Lab, Evry Val d'Essonne University, Évry, France
Hakim.ElChaoui@ibisc.univ-evry.fr
[2] LIMIARF Lab, Mohammed V University, Rabat, Morocco

Abstract. Applications such as Simultaneous Localization and Mapping (SLAM) can greatly benefit from RGB-D sensor data to produce 3D maps of the environment as well as sensor's trajectory estimation. However, the resulting 3D points map can be cumbersome, and since indoor environments are mainly composed of planar surfaces, the idea is to use planes as building blocks for a SLAM process. This paper describes an RGB-D SLAM system benefiting from planes segmentation to generate lightweight 3D plane-based maps. Our goal is to produce reduced 3D maps composed solely of planes sections that can be used on platforms with limited memory and computation resources. We present the introduction of planar regions in a regular RGB-D SLAM system and evaluate the benefits regarding both resulting map and estimated camera trajectory.

Keywords: RGB-D SLAM · Planar features · 3D Plane-based maps

1 Introduction

In recent years, mobile robots have received an increasing interest especially for indoor environment applications. In these environments RGB-D sensors offer an opportunity to significantly develop robotic navigation and interaction capabilities. Since they combine the advantages of RGB cameras with the ability to obtain geometric information, many works tend to exploit the potential of these novel sensors. For applications such as Simultaneous Localization and Mapping (SLAM), introducing RGB-D cameras allows to create three-dimensional maps in real time. Several RGB-D SLAM systems have been proposed. They can be placed in two large family: Sparse and Dense SLAM systems. Although the purpose is the same, the two approaches diverge in the modeling and processing. Sparse SLAM approaches are based on visual odometry. They use visual features correspondences with registration algorithms, as RANSAC [5] or ICP [16], to estimate and refine transformations between poses. The algorithm developed by Henry *et al.* [8] was one of the first RGB-D systems using features points to estimate camera poses and represent the environment by surfels [15]. It creates and optimizes a Graph-Based SLAM. This modeling [7] consists in constructing a graph which nodes are sensor poses and where an edge between two nodes represents the transformation (egomotion) between these poses. Such formulation enables the graph optimization step which aims to find the best nodes configuration that produces a correct topological trajectory and easier loop-closures detection when revisiting the same areas. Endres *et al.* [4] followed the same path and proposed a graph-based RGB-D SLAM which became very popular among Robotic Operating System (ROS) users due to its availability. The implementation and optimization of the pose-graph is performed by the G^2o framework [13]. To represent the environment, they used 3D occupancy grid maps generated by the OctoMapping approach [9]. This RGB-D SLAM system offers a good trade-off between the quality of pose estimation and computational cost. Indeed, sparse SLAM approaches are typically fast due to the sensor's egomotion estimation based on sparse points. In addition, such a lightweight implementation can be embedded easily on mobile robots and small devices. However, the reconstruction quality is limited to a sparse set of 3D points which leads to many redundant and repeated points in the map and lack of semantic description of the environment.

Instead, dense SLAM methods enable good pose estimation and high quality scene representation. However, they tend to drift over time and fail against scenes with poor geometric structure. To overcome high computational costs, these approaches use sophisticated equipments such as high performance graphics hardware GPU which may constrain the used platform. Dense RGB-D SLAM systems were introduced by Newcombe *et al.* in the well known Kinect Fusion [10,14]. It is a dense system integrating real-time depth measurements into a volumetric data structure to create highly detailed voxel-based maps. Camera poses are estimated by tracking only live depth data with a dense iterative closest point (ICP) algorithm. Despite the high quality maps, the algorithm fails in environments with poor structure and is restricted to small workspaces due to high memory consumption. Whelan *et al.* proposed a moving volume method [23] to overcome the restricted area problem. By moving the voxel grid with each current camera pose, real-time detailed mapping of unbounded areas became possible. Unlike voxel-based reconstruction, Keller *et al.* [12] proposed a more efficient solution. They proposed a point-based fusion representation supporting spatially extended reconstructions with a fused surfel-based model of the environment.

Recently, perceiving the geometry surrounding robots has become a research field of great interest in computer vision. For both robot planing and augmented reality applications as [1], using some geometric assumptions is a crucial prerequisite. Likewise, in current RGB-D SLAM systems researchers begin to pay a significant interest to geometric primitives in order to build three-dimensional (3D) structures. The observed geometry can be a good solution to better constrain the problem and help improve 3D reconstructions. For indoor environments, 3D planes can be relevant as they are extremely common and easily deduced from point clouds. Thus, they were introduced in several recent works. One of the earliest RGB-D SLAM approaches incorporating planes has been developed by

Trevor *et al.* [21]. They combined a Kinect sensor with a large 2D planar laser scanner to generate both lines and planes as features in a graph based representation. Data association is performed by evaluating the joint probability over a set of interpretation trees of the measurements seen by the robot at one pose. Taguchi *et al.* [20] presented a bundle adjustment system combining both 3D point-to-point and 3D plane-to-plane correspondences. Their system shows a compact representation but a slow camera tracking. This work was extended by Ataer-Cansizoglu *et al.* [2] to find point and plane correspondences using camera motion prediction. However, the constant velocity assumption used to predict the pose seems to be difficult to satisfy when using handheld camera. The RGB-D SLAM system [12] was extended by Salas-Moreno *et al.* [18] to enforce planarity on the dense reconstruction with application to augmented reality. In a recent work, Xiang and Zhang [6] proposed an RGB-D SLAM system based on planar features. From each detected 3D plane they generate a 2D image and try to extract its 2D visual features. These extracted features are back-projected on the depth image to generate 3D feature points used to estimate the egomotion with ICP. More recently, Whelan *et al.* [22] performed incremental planar segmentations on point clouds to generate a global mesh model consisting of planar and non-planar triangulated surfaces. In [11], the full representation of infinite planes is reduced to a point representation in the unit sphere $\mathbb{S}3$. This allowed to parameterize the plane as a unit quaternion and formulate the problem as a least-squares optimization of a graph of infinite planes.

In our works, we are also focused on searching alternative 3D primitives in structured indoor scenes. Especially, we use 3D planar regions to generate a reduced significant representation of the environment in sparse RGB-D SLAM systems. Indeed, an RGB-D point cloud contains 307200 points and requires 3.4 Megabytes in memory. Together with the sensor's characteristic noise, the large number of points lead to significantly redundant 3D maps when assembling several point clouds. Thus, using planar assumptions on the observed geometry can deal with sensor noise and redundant representation of the environment. This paper, based on our previous work [3], describes an RGB-D SLAM system benefiting from planes segmentation to generate lightweight 3D plane-based maps. Unlike previous work, we propose a new method for building low-cost 3D maps based on reliable estimations. As human living environments are mostly composed of planar features, such technique is suitable to overcome the sensor's weakness without using a dense approach.

Our contributions in this paper are three-fold: i. We detail our 3D plane-based SLAM system implementation. ii. We propose a 3D compact planar modeling for indoor environment representations. iii. We introduce a new protocol to evaluate the system performance over real world scenes.

In the reminder of this paper we give an overview of our system in the following section. In Sect. 3 we provide an evaluation of the qualitative and quantitative performance of our system using a handheld RGB-D sensor. Finally, we summarize and report future works in Sect. 4.

2 System Description

The schematic representation of our system is shown in Fig. 1. Our starting point was inspired by the RGB-D SLAM system introduced by Endres *et al.* [4]. Inputs are color images and depth maps (RGB-D data) provided by the Kinect sensor. Our system introduces 3D planes to estimate camera's transformations and generate a planar reconstruction of the environment by merging planes into a global map. We begin by extracting local 3D planar regions (planes models with their bounding boxes) π_l from Depth maps (See next subsection) and 2D feature points from RGB data concurrently. 2D features points belonging to detected planar regions are projected onto these planes as 3D planar feature points. These planar feature points are used to estimate the local transformation T_l between two camera poses and refine it with an iterative RANSAC. Let the global coordinates system be the first registered camera pose. In each pose addition we store the global transformation T_w leading to this coordinates system using current and previous local transformations. Then, global detected planar regions π_w can be obtained. The 3D planar map is updated either by merging detected planar regions to the existing ones (Planes set π), or by adding new planes.

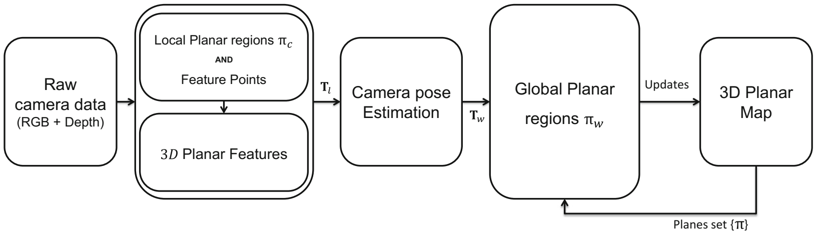

Fig. 1. Main system pipeline.

2.1 Planar Regions Detection and Representation

Planar regions detection is performed using the Point Cloud Library (PCL) [17]. We extract co-planar points sets into the full point cloud. To find the k most prominent planar regions, which contains the maximum number of 3D points (inlier points), we proceed to a multi-plane segmentation with an iterative RANSAC. At each stage, we detect the main planar region and remove its inliers from the point cloud. Then, we extract the next main planar region while the number N of its inliers is still significant (at least 700 points).

A detected planar region π is parameterized by its plane model $(n_x, n_y, n_z, d)^\top$ where $n^\top(n_x, n_y, n_z)$ is the normal vector and d the distance from the origin. A 3D point $\boldsymbol{p}(x, y, z)$ lying on this plane model satisfies the familiar equation $n_x x + n_y y + n_z z = -d$, otherwise written $n^\top \ \boldsymbol{p} = -d$.

This equation will be used to generate our 3D planar feature points detailed in the next subsection.

To represent a plane in the map, we also need its bounding box $\mathcal{B}$. When a planar region is detected in the local frame, the point cloud of its inlier points is stored and then projected into the world coordinates after the egomotion estimation. Then, we generate the matrix of the second central moments of this point cloud. Afterwards, we proceed to a Singular Value Decomposition (SVD) of this matrix in order to find the main axis vectors of the point cloud and therefore the bounding box according to these vectors.

In the sequel, each detected planar region will be formerly defined by $\pi[n, d, N, \mathcal{B}]$ encompassing the plane model (n, d), its inliers population N, and its bounding box $\mathcal{B}$. These parameters will be useful during the global planar map construction (Sect. 2.3).

2.2 3D Planar Feature Points

Studies conducted in [6,20] showed that planar primitives are safer and robust to noise which leads to more accurate pose estimation while making processing faster. Then, it is relevant to introduce planar feature points in our system since depth maps provided by the Kinect sensor are noisy and contain missing depth values. The aim is to minimize 3D points measurement errors by benefiting from 3D plane fitting. Thus, we define our 3D planar feature points as 3D feature points satisfying a detected 3D planar model. The generation of these 3D planar feature points begins with 3D planar regions detection from point clouds and visual 2D feature points extraction from RGB images concurrently. For each 2D feature point we retrieve the depth value, generate its 3D feature point and check it against detected planar regions. 3D feature points belonging to a planar region are kept and others are rejected. In addition, we perform a regularization step by projecting all remaining 3D planar feature points into their respective planar regions.

2.3 Planar-Maps Building

As mentioned before, our goal is to produce lightweight 3D planar maps which can be useful for indoor robot navigation and augmented reality applications. For low-cost applications such as small robots, this choice can be very efficient to avoid 3D point clouds representations which are highly redundant and require memory resources. Based on the detected planar regions and the estimated transformation in each added pose, our system adds and grows planar structures in the map over time. To construct the global map, planes must be represented in the same 3D coordinates system. As the planes detection is performed in local frames, registered planar regions can be transformed from the camera to world coordinates systems using the global egomotion estimation T_w. If the matrix M (Rotation R and Translation t) represents this global transformation, a 3D point $\boldsymbol{p}_w$ into the world coordinates can be found easily using its correspondent point in the camera $\boldsymbol{p}_c$ by the well known equation: $\boldsymbol{p}_w = \mathrm{R}\boldsymbol{p}_c + t$ and conversely

$\boldsymbol{p}_c = R^\top \boldsymbol{p}_w - R^\top t$. Then, a plane in the world coordinates system is defined by its normal vector and distance $\pi_w(n_w, d_w)$ with $n_w = Rn_c$ and $d_w = d_c - n_c^\top R^\top t$. Once these parameters are defined, we proceed to a plane-to-plane comparison in order to update the global map. Whenever a new planar region matches to a registered one, we update the plane by merging their plane models. Correspondence between planar regions is examined in two stages (Fig. 2). Firstly, angle between two compared plane models must not exceed a threshold set to $10\,^\circ$ and the distance limit between them is set to $5\,cm$. When this is satisfied, we check that the two planar regions overlap or are spatially close at least. This condition is added to avoid merging planes which have the same model but don't represent a continuous plane in the real world. An example of such situation can be a wall with an open door in the middle: the two parts of the wall have the same equation. Thus, using the bounding boxes of concerned planar regions we obtain the overlap between them if it exists. In case there is no overlap, we compare the minimum distance between the two bounding boxes to a distance threshold of $50\,cm$. When two planar regions match, they are merged together in a new resulting plane according to their inlier points populations. Their bounding boxes are compared and extrema are assigned to the merged planar region. If no correspondence can be found, the detected region is added to the map as a new plane. Even more than planes themselves, our map contains theoretical intersections between these planes. Planes intersections are generated using an adjacency criterion. We represent this intersection by lines and points when two or three planes intersect. This makes our map more significant and workable for other applications, and represents a first step towards a more semantic map.

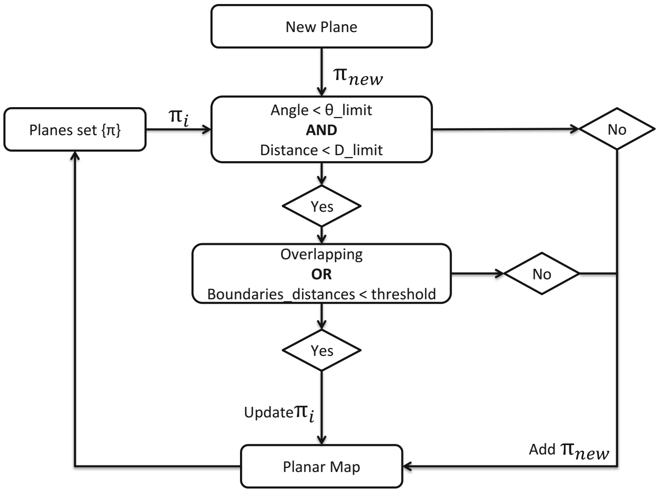

Fig. 2. Planar map building.

3 Evaluation and Results

In this section we evaluate our work with a series of experiments using RGB-D data acquired with a Kinect. We present real-time results obtained on a PC with an Intel Core i5-2400 CPU at 3.10 GHz $\times$ 4. Planes detection was performed within depth images with a plane thickness threshold set to 1.5 cm. We rejected all points exceeding this threshold. Also, only the first three main planes were considered during planar regions detection. Since planes detection produces larger planes first (in terms of population), adding further planes may introduce a lot of irrelevant small planar patches. These thresholds were chosen with preliminary experiments to provide accurate results. Experimental protocol and results concerning planes quality, map building and poses trajectory are shown in the following subsections. Unfortunately similar systems do not offer public softwares which makes comparisons impossible as we propose novel evaluations.

3.1 Planes Detection Process

In order to determine the overall performance of our system, we evaluated the influence of 3D data size on the detected planes quality. By default the Kinect images resolution is 640×480. Hence, a subsampling step seems essential to limit the plane detection process time. Also, this allows online data acquisition and processing with the Kinect 30 Hz update rate. Otherwise, the point cloud will contain 307200 points, which requires several seconds for planes detection. This is often used by sparse systems to overcome computational time. So we used a subsampling factor when creating a point cloud. This subsampling factor reduces the depth image dimensions. Then, we studied the impact of subsampled data on the quality and speed of planes detection over different distances. We performed specific experiments on a plane placed in front of the Kinect. For each distance we changed the subsampling factor and observed the detected planes. Table 1 (page 8) details results of these experiments. For each experiment, we considered planes detection runtimes, estimated distances and the number of erroneous detected extra planes. From these three columns we can deduce that the subsampling factor 4 gives a good trade-off between runtime and planes estimation quality. We can notice that planes detection quality decreases with the distance as the depth noise becomes greater than the planar region thickness threshold. As the depth noise increases with distance, planes detection over 3.5 m cannot be performed. Analysis of these results allowed us to choose subsampling data with factor 4 as it doesn't degrade the detected planes quality and enables real time processing. We also noticed that angle errors between detected planes from the same distance does not exceed 6° in any case.

3.2 3D Planar Maps

Here we show results of our planar SLAM system for a real time captured office scene. The Kinect was mounted on a movable support that enables it to be placed everywhere within the scene. The office scene contains several planes

Table 1. Impact of subsampled data on planes detection.

Real distance (m)	Subsampling factor	Estimation runtime (s)	Inliers points	Estimated distance (m)	Number of erroneous detected planes
1.5	1	2.37	234813	1.49	0
	2	0.23	58797	1.48	0
	4	**0.07**	14545	**1.49**	0
2.5	1	7.26	125107	2.44	1
	2	1.78	32320	2.41	2
	4	**0.4**	8619	2.41	**0**
3.5	1	7.57	62209	3.32	2
	2	1.72	14637	3.37	1
	4	**0.4**	1904	3.36	**0**

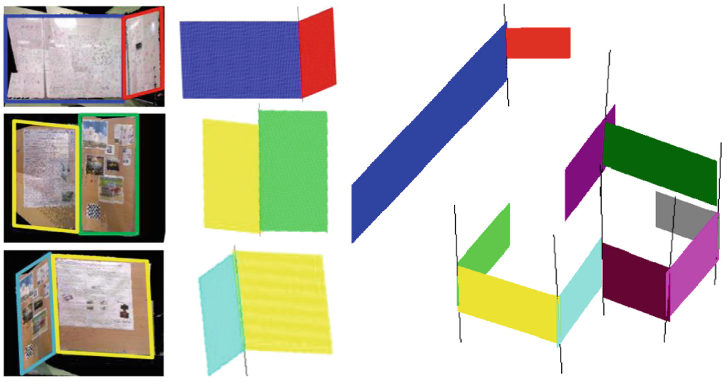

Fig. 3. Example of a 3D planar map resulting from our system. **(left)** Generated planes and their corresponding point cloud. **(right)** Our planar map with planes intersections.

with various sizes set in different locations and features several parallel and intersecting planes.

Figure 3 shows an example of our 3D plane-based map with computed intersections. Each plane in the map was generated progressively by the merging process described above. Theoretical intersections between adjacent planes was also updated whenever a plane model changed. Benefiting from this minimal representation, we generated a lightweight 3D global plane-based map unlike the usual heavyweight point-based maps.

Poses Estimation Error. In our previous work [3] we evaluated generated trajectories and runtimes of our approach against benchmark data (Sect. 4.1 Benchmark datasets). Evaluations have shown that we significantly reduced the egomotion run-time while keeping a good accuracy of the estimated trajectories over the compared approaches. Here we complet this study by introducing our novel evaluation protocol consisting in revisiting particular locations in the

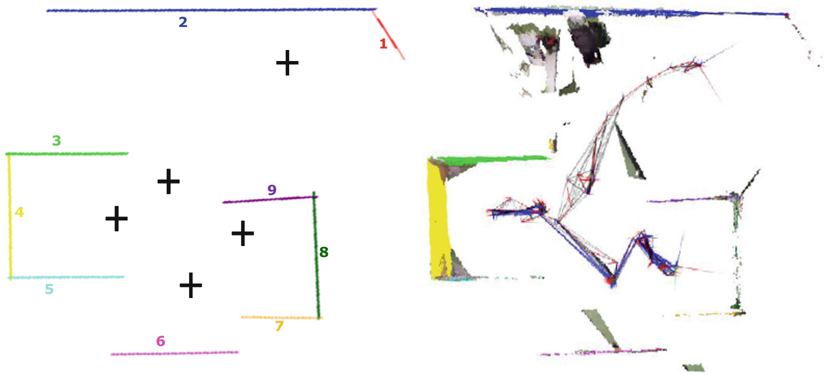

Fig. 4. Overhead view of **(right)** Point-based map with real trajectory inside and **(left)** Plane-based map with evaluation locations (crosses).

Table 2. ATE and RPE poses estimation errors.

ATE RMSE (m)	0.006 ± 0.003
Relative pose error (RPE) RMSE - Translation (m)	0.014 ± 0.002
Relative pose error (RPE) RMSE - Rotation (deg)	$1.30^\circ \pm 0.54^\circ$

scene several times and comparing stored poses. As we don't have an external tracking system, the idea is to consider the first passage at a location as the ground truth and to compare further visits poses. We chose five sparse locations in the scene (crosses in Fig. 4 **left**) and performed a series of experiments where we moved through these locations randomly. Such experiments are very appreciated as by evaluating the revisited areas we evaluate loop closures and thus the system's accuracy. Table 2 summarises the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE) proposed by Sturm *et al.* [19]. ATE computes the absolute difference between the ground truth and estimated poses after alignment and RPE measures the local accuracy of the trajectory over a fixed time interval Δ (equals 1 here). Results show poses estimation accuracy and consequently the robustness of our system. This confirms the 3D planar feature points reliability already shown in our previous work.

Evaluations of the Map. As our goal is to build planar maps which can be used in several contexts such as mobile navigation and augmented reality, it is very important to assess our map against real world scenes. Our evaluation, first time proposed, consists in comparing measurements of the generated maps with those of real scenes, namely distances and angles between planes. Tables 3 and 4 represent distances and angles estimations between numbered planes in Fig. 4 compared to the real ones in the office scene. Evaluations show that errors between real and estimated measurements do not exceeds 6 % for distances and

Table 3. Real and estimated distances between planes.

Plane i	Plane j	Real distance (Avg. ± Std. Dev.)	Estimation (Avg. ± Std. Dev.)
2	3	$1.50\,m \pm 0.02\,m$	$1.45\,m \pm 0.06\,m$
2	5	$2.82\,m \pm 0.02\,m$	$2.70\,m \pm 0.08\,m$
2	6	$3.42\,m \pm 0.02\,m$	$3.40\,m \pm 0.06\,m$
2	7	$3.20\,m \pm 0.02\,m$	$3.00\,m \pm 0.10\,m$
2	9	$1.95\,m \pm 0.02\,m$	$1.81\,m \pm 0.10\,m$
4	8	$2.95\,m \pm 0.02\,m$	$2.88\,m \pm 0.05\,m$

Table 4. Real and estimated angles between planes.

Plane i	Plane j	Real angle (Avg. ± Std. Dev.)	Estimation (Avg. ± Std. Dev.)
1	2	$56\,^\circ \pm 2\,^\circ$	$56.80\,^\circ \pm 1.5\,^\circ$
3	4	$90\,^\circ \pm 2\,^\circ$	$92.37\,^\circ \pm 2.1\,^\circ$
4	5	$90\,^\circ \pm 2\,^\circ$	$90.12\,^\circ \pm 0.7\,^\circ$
7	8	$90\,^\circ \pm 2\,^\circ$	$92.52\,^\circ \pm 1.4\,^\circ$
8	9	$90\,^\circ \pm 2\,^\circ$	$91.75\,^\circ \pm 0.9\,^\circ$

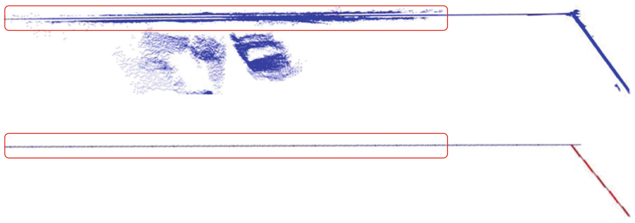

Fig. 5. Close overhead view of a planar region in the scene: **(top)** Raw point cloud representation and **(bottom)** 3D Plane-based map.

1 % for angles which is suitable for application requiring good accuracy. Hence, this lightweight map presents a good tradeoff between quality and memory consumption which is required by mobile robots and small devices.

3D Representations Comparison. Figure 5 shows a very close view of the second plane in Fig. 4 with two different 3D representations. The top row presents a point-based representation (Raw point cloud) and at the bottom we show our plane-based map which is a 3D planar representation. The point-based representation of this single plane already contains 340 thousand 3D points including many overlapping points due to assembling several point clouds. Such a representation would require a heavyweight process for the visualization of the entire map which leads to memory inefficiency. Our lightweight plane-based map is ready and much more usable for semantic labeling than raw 3D points based maps.

4 Conclusion

In this paper we proposed a simple 3D planar maps representation for RGB-D SLAM systems. We generated a 3D map based on detected 3D planes on the scene instead of the heavyweight point-based representation. During the reconstruction process, discovered planar areas are appended to the plane-based map. Moreover, all matched planes are progressively merged together in this map to give a compact representation and more information about the scene. Thus, we generated dense planar maps with significantly reduced sizes without relying on a dense approach. Besides, evaluations show that our system is able to build good quality maps while generating accurate trajectory. Also, the best tradeoff between precision and process time for planes detection have been evaluated. Once our plane-based map is available, the next step will consist in building the semantics associated to these planes such as floors or walls and planes structures such as desks or rooms which represents a first step towards semantic maps.

References

1. Ababsa, F., Mallem, M.: Robust camera pose tracking for augmented reality using particle filtering framework. Mach. Vis. Appl. **22**(1), 181–195 (2011)
2. Ataer-Cansizoglu, E., Taguchi, Y., Ramalingam, S., Garaas, T.: Tracking an rgb-d camera using points and planes. In: 2013 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 51–58, December 2013
3. Elghor, H.E., Roussel, D., Ababsa, F., Bouyakhf, E.H.: Planes detection for robust localization and mapping in RGB-D SLAM systems. In: 2015 International Conference on 3D Vision, 3DV 2015, pp. 452–459, October 2015
4. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3D mapping with an RGB-D camera. IEEE Trans. Robot. **30**(1), 177–187 (2014)
5. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
6. Gao, X., Zhang, T.: Robust rgb-d simultaneous localization and mapping using planar point features. Robot. Auton. Syst. **72**, 1–14 (2015)
7. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based SLAM. IEEE Intell. Transp. Syst. Mag. **2**(4), 31–43 (2010)
8. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. Int. J. Robot. Res. (IJRR) **31**(5), 647–663 (2012)
9. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: an efficient probabilistic 3d mapping framework based on octrees. Auton. Robots **34**(3), 189–206 (2013)
10. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST 2011, pp. 559–568. ACM, New York, October 2011
11. Kaess, M.: Simultaneous localization and mapping with infinite planes. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 4605–4611, May 2015

12. Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., Kolb, A.: Real-time 3d reconstruction in dynamic scenes using point-based fusion. In: 2013 International Conference on 3D Vision - 3DV 2013, pp. 1–8. IEEE Computer Society, Washington (2013)
13. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: a general framework for graph optimization. In: IEEE International Conference on Robotics and Automation (ICRA 2011), pp. 3607–3613. IEEE, May 2011
14. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: real-time dense surface mapping and tracking. In: 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2011), ISMAR 2011, pp. 127–136. IEEE Computer Society, October 2011
15. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, pp. 335–342 (2000)
16. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, (3DIM 2001), pp. 145–152, May 2001
17. Rusu, R., Cousins, S.: 3d is here: point cloud library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA 2011), pp. 1–4. IEEE, May 2011
18. Salas-Moreno, R.F., Glocken, B., Kelly, P.H.J., Davison, A.J.: Dense planar slam. In: 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 157–164, September 2014
19. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), pp. 573–580, October 2012
20. Taguchi, Y., Jian, Y.D., Ramalingam, S., Feng, C.: Point-plane SLAM for hand-held 3D sensors. In: IEEE International Conference on Robotics and Automation (ICRA 2013), pp. 5182–5189, May 2013
21. Trevor, A.J.B., Rogers, J.G., Christensen, H.I.: Planar surface slam with 3d and 2d sensors. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 3041–3048, May 2012
22. Whelan, T., Ma, L., Bondarev, E., de With, P., McDonald, J.: Incremental and batch planar simplification of dense point cloud maps. Robot. Auton. Syst. **69**(C), 3–14 (2015)
23. Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J.J., McDonald, J.: Real-time large-scale dense rgb-d slam with volumetric fusion. Int. J. Robot. Res. **34**(4–5), 598–626 (2015)