sensors

MDPI

Article

Point-Plane SLAM Using Supposed Planes for Indoor Environments

Xiaoyu Zhang [1], Wei Wang [1,*], Xianyu Qi [1], Ziwei Liao [1] and Ran Wei [2]

[1] Robotics Institute, Beihang University, Beijing 100191, China
[2] Beijing Evolver Robotics Technology Co., Ltd., Beijing 100192, China
* Correspondence: wangweilab@buaa.edu.cn; Tel.: +86-010-8231-4554

Received: 30 July 2019; Accepted: 29 August 2019; Published: 2 September 2019

Abstract: Simultaneous localization and mapping (SLAM) is a fundamental problem for various applications. For indoor environments, planes are predominant features that are less affected by measurement noise. In this paper, we propose a novel point-plane SLAM system using RGB-D cameras. First, we extract feature points from RGB images and planes from depth images. Then plane correspondences in the global map can be found using their contours. Considering the limited size of real planes, we exploit constraints of plane edges. In general, a plane edge is an intersecting line of two perpendicular planes. Therefore, instead of line-based constraints, we calculate and generate supposed perpendicular planes from edge lines, resulting in more plane observations and constraints to reduce estimation errors. To exploit the orthogonal structure in indoor environments, we also add structural (parallel or perpendicular) constraints of planes. Finally, we construct a factor graph using all of these features. The cost functions are minimized to estimate camera poses and global map. We test our proposed system on public RGB-D benchmarks, demonstrating its robust and accurate pose estimation results, compared with other state-of-the-art SLAM systems.

Keywords: SLAM; RGB-D camera; factor graph; planes; plane edges; structural constraints; indoor environments

1. Introduction

Simultaneous localization and mapping (SLAM) develops quickly in recent years and becomes a fundamental problem for various applications including mobile robots, augmented and virtual reality. Various sensors can be used for SLAM such as laser-range finders [1,2] and cameras [3]. Laser-range finders provide accurate information about the environments but they are too expensive to be widely adopted. Cameras can also provide abundant information, which are much cheaper. With the availability of cheap RGB-D cameras [4,5], the depth of the scenes can also be measured more easily, especially for indoor environments.

Most of the existing methods for SLAM are based on a collection of points and use points to describe the scenes and estimate the camera poses. Points can be described by simple mathematical expressions and applied in both indoor and outdoor environments. But these methods encounter various problems in practice application, such as low-texture environments and changing light. Besides, the correct data association is also a challenge for point-based SLAM to obtain reliable estimation results. Direct methods are based on image intensities, which can be affected by changing light or viewing angles. Feature-based methods generally search corresponding points based on descriptors, so their results depend on the reliance of detecting and matching of feature points. The error from points measurement noise and data association will accumulate, especially in large scenes. These problems are hard to solve using only points.

For indoor environments, there are lots of other high-level features, such as lines and planes. Indoor environments are also common working scenes for mobile robots. These high-level features ensure faster and more accurate data association, which can be extracted easily using RGB-D cameras. The planes calculating from many points are more robust and accurate, because of less affection from measurement noise. Therefore using these high-level features helps to improve the performance of SLAM. In indoor environments, there are various man-made objects and structures, which have lots of parallel and perpendicular planes. Using these kinds of structural constraints can also help to achieve a long-term association for planes, resulting in smaller accumulated error. The plane is usually described as an infinite plane in mathematics. However, the real planes in working environments have limited size, contours or edges. Therefore, these features can also be exploited to add constraints for robust pose estimation.

In this paper, we propose a SLAM system using both points and planes to achieve robust and accurate estimation results. Using a RGB-D camera, we detect and match feature points in RGB images and generate point clouds from depth image to extract planes. Unlike other plane-based SLAM using infinite planes, we try to make use of plane edges. For indoor environments, a plane edge is generally an intersecting line of two perpendicular planes. In order to add the constraints of plane edges, we calculate and generate their perpendicular planes even when they are not seen. We also use contour points of planes to achieve robust data association. Then we use all of these points and planes to solve the poses of the camera and generate a map consisting of points and planes. Besides, we add parallel and perpendicular constraints for planes, which help reduce drift errors in indoor environments.

In summary, the contributions of our work are as follows:

- We exploit plane edge constraints by generating supposed perpendicular planes from them.
- We achieve robust data association for planes using their contour points.
- We add perpendicular and parallel constrains for planes, which reduce drift errors in indoor environments.
- We evaluate our proposed system on public datasets and achieve state-of-the-art performance, which also performs nearly in real time.

2. Related Work

Many different SLAM algorithms have been proposed in recent years and most of them formulate SLAM problem as a nonlinear least-squares problem [6]. The point-based SLAM tracks features across frames and builds a global map consisting of points. ORB-SLAM [7] tracks ORB feature points and uses reprojection error to estimate camera poses. Direct methods [8] use intensities error to track poses. Some other point-based SLAM methods can also obtain a sparse map [9], semi-dense map [10] or even dense map [11,12]. But all of these works may have difficulty in data association and work poorly in low-texture environments or large scenes.

In recent years, planes are also exploited to refine the performance of SLAM algorithm. Some earliest works [13,14] add planes into the extended Kalman filter (EKF) state vectors, which are computational cost because of the growing size of the dense covariance matrix [15]. Therefore, they are limited to some small scenes. Gostar et al. [16] also discuss the transition model of plane features. Taguchi et al. [17] present a framework for registration combining points and planes. CPA-SLAM [18] proposes a novel formulation for tracking camera motion using global planes in an expectation-maximization (EM) framework. Although they use soft labeling to reduce the effect of incorrect plane association, it can still be wrong in global optimization. Our proposal method for hard labeling works well to deal with these problems. Kaess et al. [19] introduce a minimal representation for infinite planes which is suitable for the least-squares estimation without encountering singularities. They also develop a fast dense planar SLAM algorithm [20]. EFs [21] proposes a novel method of reformulation of plane estimation and optimizing trajectories without explicit parametrization of planes. All of these works mentioned above use infinite planes of three degrees of freedom to simplify the representation of real planes. They may work poorly in some scenes, where the plane features are not plentiful. Considering the limited size of real planes, our work exploits the edges of planes to add more useful constraints, achieving more accurate and robust estimation. Besides, previous works ignore the structural constraints for planes, which are very useful for indoor environments.

In indoor environments, spatial structure is utilized to help simplify pose estimation or even make it more robust and accurate. Yang et al. [22] propose pop-up 3D plane model to generate plane landmark measurements in SLAM. Most of the indoor scenes are based on the Cartesian coordinate system, defined as Manhattan World (MW) [23]. Zhou et al. [24] utilize mean-shift to track dominant directions of MW and achieve drift-free rotation by decoupling the estimation of rotation and translation. Some other works [25–27] also exploit planes of MW to estimate drift-free rotation. These algorithms work well in some specific scenes, but they are also easy to fail because the MW assumption is not valid for some scenes. They give us an idea to use parallel or perpendicular constraints instead of three dominant directions, which can work in more scenes in indoor environments.

3. Proposed Methods

3.1. System Overview

In this subsection, we provide an overview of our proposed point-plane SLAM using supposed planes from edges, which is shown in Figure 1. Like other modern SLAM systems, ours can also be divided into two functional parts: (1) frond-end, the tracking part extracts and matches features for new captured frame, and estimates the camera pose by minimizing the error function constituted by the tracked features in the map; (2) back-end, the map management part estimates and optimizes landmarks in the environment. The theories for point-based SLAM are thorough enough, so we augment publicly available ORB-SLAM2 [7] RGB-D implementation to build our system, and focus on exploiting planes.

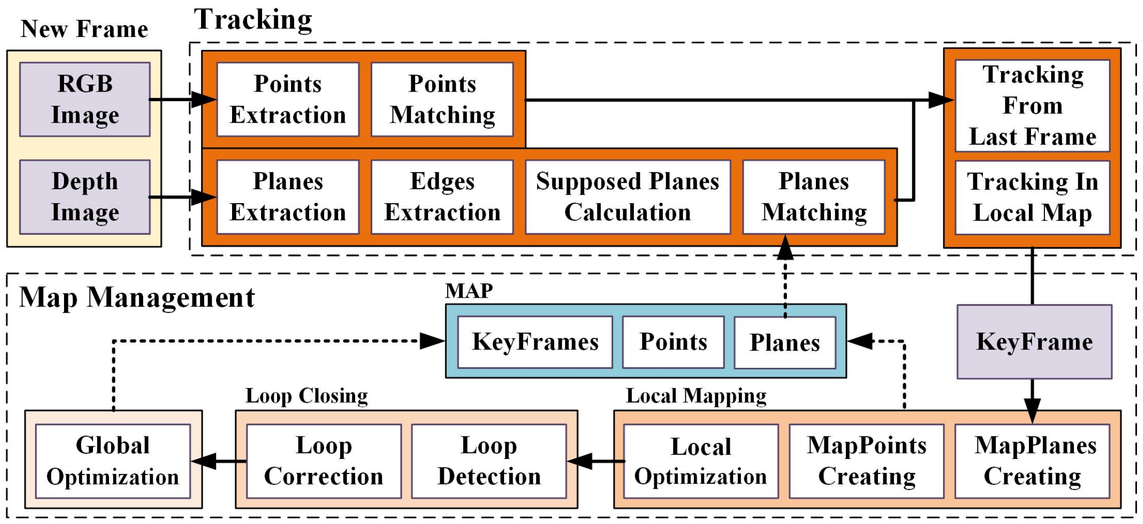


Figure 1. The overview of our proposed point-plane simultaneous localization and mapping (SLAM) using supposed planes from edges: the RGB-D camera provides RGB images and depth images as inputs of the whole SLAM system. The front-end tracks camera pose using matched points and planes. The back-end constructs and updates a consistent map consisting of keyframes, points and planes.

Global Map. The global map consists of a set of keyframes and detected landmarks, including both points and planes. The keyframes contain observed feature points and their descriptors, observed planes and supposed planes. For a point landmark, we store a list of observations and representative descriptor. Besides observations, a plane landmark also contains its contour points, edge lines and corresponding parallel or perpendicular planes, which will be explained in Section 3.3.

Tracking. The RGB-D camera (for example, Microsoft Kinect v2) provides RGB images and depth images. We extract ORB features [28] from RGB images and match them by descriptors. From depth images, we construct 3D point clouds and then extract planes. For indoor environments, to achieve more accurate and robust implementation, we also detect edges of the plane to calculate supposed planes and add constraints between parallel or perpendicular planes. With matched points and planes from the last frame or local map, the camera pose can be estimated.

Map Management. From keyframes, a local map consisting of plans and points is constructed and updated during the local mapping. We also process a global optimization after loop closing to construct a consistent global map. The loops are detected using the bag of words based on ORB features.

3.2. Preliminaries

We represent the pose of the frame k with respect to the world coordinate system w by $\mathbf{T}_{kw} \in \mathbf{SE}(3)$, which is also a rigid transformation that transforms a 3D point $\mathbf{P}_w$ from the world to the camera coordinate system:

$$\mathbf{P}_k = \mathbf{T}_{kw}\mathbf{P}_w \tag{1}$$

The point in Equation (1) is represented by homogeneous coordinates $\mathbf{P} = (p_1, p_2, p_3, p_4)^\top \in \mathbb{P}^3$, and the corresponding Euclidean point is $\mathbf{p} = (p_1/p_4, p_2/p_4, p_3/p_4)^\top \in \mathbb{R}^3$. When a 3D point $\mathbf{p}$ is observed by the camera, there is a corresponding 2D pixel $\mathbf{u} = (u, v)^\top$ in the image. Here, u and v define the position of the pixel in the image. For aligned RGB and depth images, the same point locates at the same position. The projection of $\mathbf{p}$ onto the image is $\mathbf{u} = \rho(\mathbf{p})$, and the back-projection is $\mathbf{p} = \rho^{-1}(\mathbf{u})$.

We parametrize planes using the Hessian form $\boldsymbol{\pi} = \left(\mathbf{n}^\top, d\right)^\top$, where $\mathbf{n} = \left(n_x, n_y, n_z\right)^\top$ is the unit vector representing the plane's orientation and d is the distance of the plane from the origin [29]. A point $\mathbf{p}$ lies on the plane $\boldsymbol{\pi}$ gives:

$$\mathbf{n}^\top\mathbf{p} + d = 0 \tag{2}$$

Similarly, a plane in the world frame can also be transformed into the camera frame:

$$\boldsymbol{\pi}_k = \mathbf{T}_{kw}^{-\top}\boldsymbol{\pi}_w. \tag{3}$$

3.3. Plane Features

We construct a global map that consists of all plane features in the scenes. Every plane is segmented from organized point clouds generated from depth images. Considering the limited size of real planes, we also exploit constraints of plane edges. We calculate a supposed plane from the plane edge, which may also be observed by other frames. We try to match the plane with all planes in the map and use plane-to-plane constraints to estimate and refine the camera poses.

3.3.1. Plane Segmentation

The RGB-D camera provides RGB images and aligned depth images. In a depth image, each pixel relates to a distance between the image plane and the corresponding object in the RGB image. So we can recover the structure using the camera model to back-project the pixel and we use the pinhole camera model [29] in our work:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = d \begin{pmatrix} f_x^{-1} & 0 & -c_x f_x^{-1} \\ 0 & f_y^{-1} & -c_y f_y^{-1} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \tag{4}$$

where $\mathbf{u} = (u, v)^\top$ is the valid pixel in depth image and d is the value in the depth image. $\mathbf{p} = (x, y, z)^\top$ is the corresponding 3D point. f_x and f_y are focal length of the camera, and $\left(c_x, c_y\right)$ is the camera center coordinate.

The point clouds generated from depth images are organized, having an image like grid structure. Organized structure enables fast plane segmentation from the point cloud. We follow the work of [30], which segments point clouds from RGB-D data in near real-time. In this process, we can also obtain the contour of the segmented plane. The contour will be useful for calculating supposed planes and obtaining robust data association of planes.

3.3.2. Supposed Plane Serving as Edge Constraint

As we have described, the plane can be parametrized by simple mathematical expression $\pi = \left(\mathbf{n}^\top, d\right)^\top$. π represents an infinite plane, which has three degrees of freedom. It means the plane can slide along the vertical direction of its normal. But the real planes in the scenes are not infinite and they have boundaries and edge lines. Therefore, using only these four parameters without edges loses other information from the real planes. Moreover, usually few planes can be observed in one frame, resulting in insufficient constraints for pose estimation. So we need to exploit more planes or constraints from their edges to estimate camera poses in real scenes.

In indoor environments, most man-made objects or structures have regular shapes, especially those objects that have large enough plane features. Therefore, it becomes easy to extract edge lines from these segmented planes. The edge of a plane can also be seen as an intersecting line with another plane. Besides, these two planes are generally perpendicular to each other. To add constraints from plane edges, we calculate and generate a supposed perpendicular plane from every edge line, instead of adding line-based constraints directly. For every captured frame, we not only segment planes from the depth image, but also calculate such supposed planes if there are valid plane edges. Note that, supposed planes may be also observed in other frames.

When segmenting planes from organized point clouds, we can also acquire the contour of segmented planes. We extract edge lines from contour using RANSAC [31]. If the inliers are sufficient (more than 15 percent of the contour points in our experiments), the extracted line is valid. Then we examine the position of the line to avoid the border of the image. We also examine the points near the lines to remove those lines extracted from shadow borders. A valid edge line can be represented by

$$\mathbf{l} = \left(\mathbf{p}_l^\top, \mathbf{n}_l^\top\right)^\top = \left(p_x, p_y, p_z, n_x, n_y, n_z\right)^\top, \tag{5}$$

$\mathbf{p}_l^\top$ is a point on this line, and $\mathbf{n}_l^\top$ is the direction vector of the line.

The supposed perpendicular plane is calculated using the plane and its edge line. Having the representation of the plane $\pi_i = \left(\mathbf{n}_i^\top, d_i\right)^\top$ and its edge line $\mathbf{l}_j = \left(\mathbf{p}_j^\top, \mathbf{n}_j^\top\right)^\top$, the supposed plane is:

$$\pi_{supposed} = \begin{pmatrix} \mathbf{n}_{ij} \\ d_{ij} \end{pmatrix} = \begin{pmatrix} \mathbf{n}_i \times \mathbf{n}_j \\ -\mathbf{n}_{ij}^\top \mathbf{p}_j \end{pmatrix} \tag{6}$$

The process of generating supposed planes is shown in Figure 2. It is a frame from the sequence 'freiburg3_cabinet' of Technical University of Munich (TUM) RGB-D [32] dataset. There is a cabinet in the image, and three planes of the cabinet are extracted. We know there are five planes which can be observed, except for the bottom plane. But only three planes can be observed at most in one frame, although we can imagine the other two planes. To exploit more constraints of such common man-made objects, we suppose planes (the green planes in Figure 2c) from the plane edges. We remove those repeating planes. When the camera goes to observe in the opposite direction, these supposed planes will be observed.

We can treat supposed planes as ordinary planes, like those observed by the camera directly. Plane landmarks are also created from supposed planes and may be observed by other frames. Therefore, supposed planes increase the number of planes and add constraints from plane edges.

Besides, supposed planes have a perpendicular orientation to the corresponding planes. Planes of different orientations provide more sufficient constraints for accurate pose estimation.

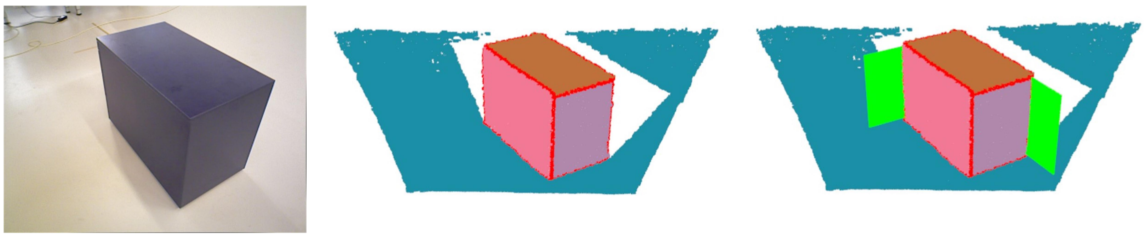

(**a**) Cabinet (**b**) Planes extracted from the cabinet (**c**) Supposed planes

Figure 2. The process of generating supposed planes: (**a**) Cabinet. A frame from sequence 'freiburg3_cabinet' of TUM RGB-D dataset. There is only one cabinet in the image. (**b**) Planes extracted from the cabinet. Three planes of the cabinet are extracted. Different colors denote different planes. The red points are those belong to valid plane edges. (**c**) Supposed planes. The green planes denote supposed planes generated from plane edges. Other repeating supposed planes are removed.

3.3.3. Data Association

Fast and accurate data association is the guarantee for robust and accurate pose estimation. Feature points can be associated with descriptors. For planes, previous work [18,19] find their correspondences just by normal $\mathbf{n}^\top$ and distance d. This method does work for simple environments. But it also depends on an accurate pose estimation of the camera, which is not often satisfied, especially when the current pose is predicted from the previous pose. When the angle of the plane has some noise, the distance usually fluctuates largely, leading to association failure.

We implement a novel data association algorithm for matching planes. First, we find those intersecting planes on the map. In indoor environments, intersecting planes are corresponding planes with a small-angle difference because of noise, or different planes (usually perpendicular planes) with a large-angle difference. Therefore, we just need to check the angle of these intersecting planes to find correspondence, and the angle threshold can be a little larger (30° in our experiments).

To find intersecting planes in the global map, we calculate the distances from the points on the observed planes to the plane landmarks $\pi_i = \left(\mathbf{n}_i^\top, d_i\right)^\top$. We only examine the points $\mathbf{p}_j$ from the contour C_m acquired during plane segmentation to reduce the calculating amount:

$$s = \min\{|\mathbf{n}_i^\top \mathbf{p}_j + d_i|\}, \mathbf{p}_j \in C_m. \tag{7}$$

If s is smaller than the distance threshold (0.1 m in our experiments), these two planes are considered intersecting planes. Then we examine the angle of these two planes to determine the corresponding plane. For every plane observation, we try to find a corresponding plane with the smallest s in the global map. If a plane observation fails to be associated, it is added to the global map as a new plane landmark.

Besides corresponding planes, we also exploit parallel and perpendicular planes. For every plane, we find a parallel plane if their intersection angle is small enough (10° in our experiments) but the distance s between them is large (larger than 0.1 m in our experiments). If there are several planes, we choose the plane having the smallest intersection angle. Similarly, we find a perpendicular plane if the angle between them is large enough (80° in our experiments).

Therefore, for every plane observation, we try to find one corresponding plane, one parallel plane and one perpendicular plane on the global map.

3.4. Tracking Using Points and Planes

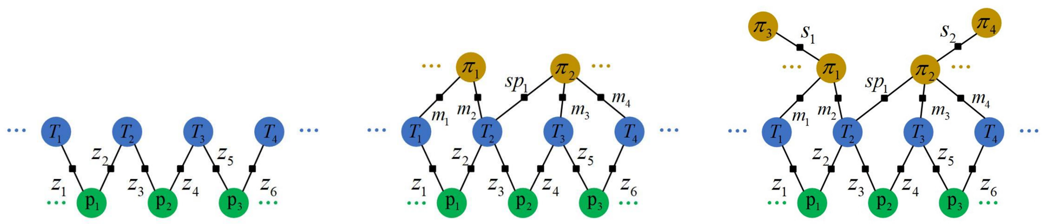
It is well known that SLAM problem can be represented as a factor graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ [33]. The vertices $\mathcal{V}$ represent the variables to estimate, such as camera poses and landmarks. The edges $\mathcal{E}$ between vertices represent the constraints. The factor graph enables an insightful visualization of SLAM problem. The process of solving SLAM problem is to construct such a factor graph and minimize the errors of all involved factors.

3.4.1. Factor Graph Containing Points and Planes

As mentioned before, we extract ORB features in the RGB image for the current frame and find matched point landmarks using ORB descriptors. A simple point-based SLAM can be represented as a factor graph in Figure 3a. The factors are constraints of the reprojection error:

$$f_z\left(\mathbf{p}_w, \mathbf{T}_{cw}\right) = \left\|\mathbf{u}_m - \rho\left(\mathbf{T}_{cw}\mathbf{P}_w\right)\right\|_{\mathbf{\Sigma}_z}, \tag{8}$$

$\mathbf{T}_{cw}\mathbf{P}_w$ is the 3D point in the camera coordinate system and $\mathbf{u}_m$ is the corresponding pixel in the current frame. $\|\mathbf{x}\|_{\mathbf{\Sigma}}$ is the mahalanobis norm, which equals $\mathbf{x}^\top\mathbf{\Sigma}^{-1}\mathbf{x}$, and $\mathbf{\Sigma}$ is the corresponding covariance matrix.



(**a**) point-based SLAM (**b**) point-plane-based SLAM (**c**) SLAM adding structure constraints

Figure 3. Factor graphs for SLAM. Circles denote vertices and black squares denote factors. (**a**) A simple point-based SLAM. The blue circles denote the keyframe poses and the green circles are point landmarks. (**b**) A point-plane-based SLAM. Yellow circles denote the plane landmarks. *sp* means supposed planes in the frame. (**c**) A point-plane-based SLAM adding structural constraints. More planes are included in the local map to add parallel or perpendicular constraints.

Likewise, planes can also be added into the factor graph as landmarks. As shown in Figure 3b, m means the direct observation of planes and sp denotes the plane is supposed in the frame. Therefore, π_2 is a supposed plane in frame T_2, but it can be observed in frame T_3 and T_4.

Now we need to define the factor connecting the vertices $\mathcal{V}(T)$ and $\mathcal{V}(\boldsymbol{\pi})$. Notice that the Hessian form $\boldsymbol{\pi} = \left(\mathbf{n}^\top, d\right)^\top$ is an over-parameterization of planes, because a 3D plane has only three degrees of freedom. Therefore, the Hessian form requires extra constraints to ensure the unit length of the plane normal vector, adding additional computation in optimization. To overcome this problem, we choose minimal parameterization of planes in optimization $\boldsymbol{\tau} = (\phi, \psi, d)$, where ϕ and ψ are the azimuth and elevation angle of the normal respectively,

$$\tau = q(\boldsymbol{\pi}) = \left(\phi = \arctan\frac{n_y}{n_x}, \psi = \arcsin n_z, d\right)^\top, \tag{9}$$

The azimuth and elevation should be restricted in $(-\pi, \pi]$ to avoid the singularities of the minimal representation in optimization. Now we can define the factor for planes:

$$f_m\left(\boldsymbol{\pi}_w, \mathbf{T}_{cw}\right) = \left\|q\left(\boldsymbol{\pi}_m\right) - q\left(\mathbf{T}_{cw}^{-\top}\boldsymbol{\pi}_w\right)\right\|_{\mathbf{\Sigma}_m}, \tag{10}$$

where $\mathbf{T}_{cw}^{-\top}\boldsymbol{\pi}_w$ is the 3D plane in the camera coordinate system. $\boldsymbol{\pi}_m$ is the corresponding plane observation in the current frame. Note that the covariance should be a little larger for those supposed planes, like sp_1 in Figure 3b.

Besides, we also add the structure factors $\mathcal{E}(S)$ for indoor environments. Structure factors, parallel or perpendicular constraints between planes, add more edges between planes, as shown in Figure 3c. The structure factors only add normal constraints of planes. For parallel constraints:

$$f_{sp}\left(\boldsymbol{\pi}_w, \mathbf{T}_{cw}\right) = \left\| q_{\mathbf{n}}\left(\boldsymbol{n}_m\right) - q_{\mathbf{n}}\left(\mathbf{R}_{cw}\boldsymbol{n}_w\right)\right\|_{\boldsymbol{\Sigma}_{sp}}, \tag{11}$$

where $q_{\mathbf{n}}$ means the azimuth and elevation angle in Equation (9) and $\boldsymbol{n}$ is the normal of the plane. $\mathbf{R}_{cw}$ is the rotation of the current frame. Note that, if normal vectors point to opposite directions, we need to rotate them to the same direction first.

Similarly, the factors for perpendicular planes:

$$f_{so}\left(\boldsymbol{\pi}_w, \mathbf{T}_{cw}\right) = \left\| q_{\mathbf{n}}\left(\mathbf{R}_{\perp}\boldsymbol{n}_m\right) - q_{\mathbf{n}}\left(\mathbf{R}_{cw}\boldsymbol{n}_w\right)\right\|_{\boldsymbol{\Sigma}_{so}}. \tag{12}$$

The only difference is an additional rotation matrix $\mathbf{R}_{\perp}$ to rotate the normals to the same direction.

3.4.2. Pose Estimation

For every new captured frame, we extracted feature points from the RGB image, planes and supposed planes from the depth image. If the last frame is tracked successfully, we use a constant velocity model to predict the pose of the current frame and search point landmarks observed in the last frame. Corresponding planes, parallel and perpendicular planes can also be searched in the global map. Because there are not too many planes in scenes, this search simple method is efficient enough. With matched points and planes, the current pose is then optimized. We also try to search more point correspondences in the local map. The local map contains several keyframes sharing point and plane landmarks with the current frame. The current pose is finally optimized with all points and planes in the local map.

With tracked points and planes, pose $\mathbf{T}_{cw}$ can be computed by solving:

$$\mathbf{T}_{cw} = \underset{T_{cw}}{\arg\min}\left(\sum H_z\left(f_z\right) + \sum H_m\left(f_m\right) + \sum H_{sp}\left(f_{sp}\right) + \sum H_{so}\left(f_{so}\right)\right), \tag{13}$$

where $H(x)$ is the Huber robust cost function and f are factors described in Section 3.4.1. We use the Levenberg–Marquadt method implemented in g2o [34] to solve this equation. The Jacobians of these components in Equation (13) are described detailedly in Appendix A. To remove bad observations, we check the error value of every factor after optimization and delete those larger than the threshold.

3.4.3. Keyframe Decision

The keyframes are used to construct the local and global map. Besides, the loop detection and relocalization are implemented based on keyframes. We utilize the new keyframe criteria in [7]. If a successfully tracked frame observes a new plane landmark or tracks less than 90% points of the last keyframe, it will be labeled as "keyframe". If N_f frames have passed from the last keyframe, it will also be constructed as a keyframe. We set N_f smaller than the output frequency of the RGB-D camera to ensure at least one keyframe will be inserted in one second. Those redundant keyframes will also be deleted later according to the number of landmarks observed.

3.5. Map Management

Map management is the back-end part of the whole system. It constructs a consistent global map and the camera poses are further optimized.

3.5.1. Local Mapping

Every time the tracking part inserts a keyframe, new point landmarks are created by triangulation. A new plane landmark is also created if a new plane is observed or supposed. The local optimization optimizes the camera poses and landmarks on the local map. The local map contains a set of keyframes sharing landmarks with the currently processed keyframe, all of the landmarks observed by these keyframes and the parallel or perpendicular planes in the map. Other keyframes that also observe the landmarks in the local map are also included in the local optimization to provide sufficient constraints but remain fixed. The cost function is also created using all of the factors described in Section 3.4.1.

After local optimization, we delete those redundant keyframes, whose 90% of the observed points and all of the observed planes have been observed by other keyframes. The bad points and planes that have large errors are also deleted.

3.5.2. Loop Closing and Gobal Optimization

We compute the bags of words representation of every keyframe to detect loops. The loop detection part is implemented using DBow2 [35]. Once a loop is detected, we perform a pose graph optimization. All the camera poses, points and planes in the global map will also be optimized in the global optimization. The global optimization uses the same factors as local optimization but has all the vertices in the global map. The drift error is reduced after the global optimization.

4. Experiments

We evaluate our proposed SLAM system using the benchmarks TUM RGB-D dataset [32] and ICL-NUIM deteset [36]:

- TUM RGB-D dataset is a famous benchmark for evaluating vSLAM/VO systems. It contains a large set of image sequences recorded from a RGB-D camera. The RGB and depth images are captured with a 640×480 resolution at the video frame rate (30 Hz). The ground truth camera poses are also provided. The dataset covers various indoor scenes and we choose those having obvious plane landmarks to evaluate our SLAM system.
- ICL-NUIM dataset is a synthetical benchmark. The images are captured within synthetically generated indoor environments, including living room and office scenes. These scenes are suitable for our point-plane SLAM system. It also provides ground truth like the TUM RGB-D dataset, which is convenient to evaluate the results.

For implementation, our system augments the RGB-D variant of ORB-SLAM2 [7]. We rely on the underlying ORB-SLAM2 for points extraction and matching. We also use the methods in ORB-SLAM2 to maintain a local map and detect loops. The focus of our implementation is on the plane extraction, supposed plane calculation, plane matching, and pose estimation using points and planes. We construct a global map consisting of points and planes, including supposed planes. All experiments run on a laptop computer with i7-7700HQ 2.80GHz CPU, 16GB RAM, without GPU.

We compare our proposed SLAM system with some other RGB-D SLAM systems. ORB-SLAM2 [7] is the state-of-the-art feature-point based visual SLAM system and it has a RGB-D implementation. L-SLAM [25] is a RGB-D SLAM system using planes and MW constraints. Note that we test ORB-SLAM2 using the open-source code provided by the author and we include the results of L-SLAM from [25] directly. To demonstrate the benefit of supposed planes, we compare the results of different versions of our proposed SLAM system, which is point-plane SLAM (PP), point-plane SLAM using structural constraints (PP+S) and point-plane SLAM adding supposed planes (PP+SS).

4.1. ICL-NUIM Dataset

The ICL-NUIM dataset contains several sequences from two kinds of scenes, office and living room, as shown in Figure 4. Four sequences are recorded in each scene. We run experiments on all of these sequences.

(**a**) Living Room

(**b**) Office Room

Figure 4. Scenes contained in the ICL-NUIM dataset. (**a**) The living room contains sofa, chairs table and some other common man-made objects. (**b**) The office room contains several tables, computers and pictures.

We use the root mean square error (RMSE) of the absolute trajectory error (ATE) to evaluate the performance of different SLAM systems. We report the results in Table 1. The smallest error for every sequence is labeled by the bold number. The comparison of the RMSE is also shown in Figure 5.

Table 1. Evaluation results of translation absolute trajectory error (ATE) root mean squared error (RMSE) (unit: m) on ICL-NUIM dataset. PP, PP+S, PP+SS denote proposed point-plane SLAM, point-plane SLAM using structural constraints, point-plane SLAM using structural constraints and supposed planes, respectively. Bold numbers represent the best performances.

Sequence	ORB-SLAM2	L-SLAM	PP	PP+S	PP+SS	Frames
living_room_0	0.008578	0.012	0.008680	**0.007982**	0.008023	1509
living_room_1	0.200787	0.027	0.022113	0.013122	**0.009793**	966
living_room_2	0.031458	0.053	0.027399	0.024423	**0.019249**	881
living_room_3	0.016210	0.143	0.013832	0.013367	**0.012473**	1241
office_room_0	0.063775	0.02	0.038052	0.021606	**0.019861**	1508
office_room_1	0.084720	**0.015**	0.023280	0.022578	0.022546	966
office_room_2	0.030912	0.026	0.025042	0.024472	**0.022009**	881
office_room_3	0.034402	**0.011**	0.023617	0.019084	0.018483	1241
average	0.054680	0.038023	0.022517	0.017508	**0.016106**	

We first add plane-based constraints only (PP). Because there are enough planes in these indoor sequences, these additional useful constraints already help to improve the accuracy of the camera poses. When there are only a few feature points can be tracked, like sequence 'living_room_1', point-based SLAM (ORB-SLAM2) performs poorly. But the planes improve the performance of the pose estimation. The planes in these indoor scenes are usually parallel or perpendicular, therefore these structural constraints (PP+S) improve the accuracy further.

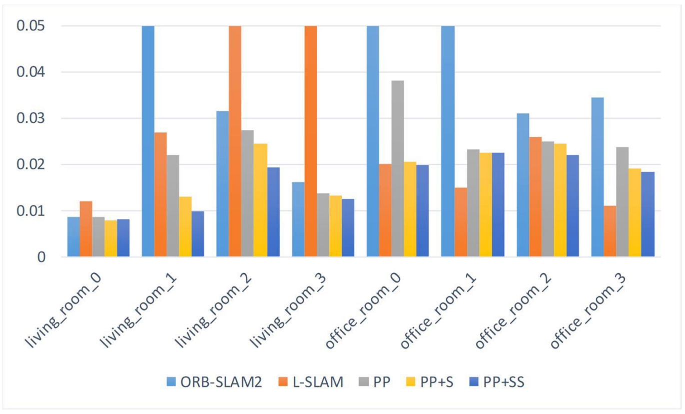
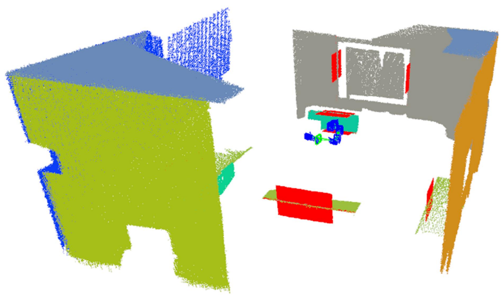


Figure 5. The comparison of the root RMSE on ICL-NUIM dataset.

Besides, we also add supposed planes (PP+SS). Table 2 shows the average number of observed planes for every frame. In some sequences, such as 'living_room_0' and 'office_room_1', the estimated trajectories are similar to those of 'PP+S', because few supposed planes can be utilized. In these sequences, the plane features are generally extracted from the wall, or planes from the furniture are too small to be extracted because the camera is far from them. Therefore, only a small number of supposed planes can be exploited. For other sequences, the supposed planes generated in the tracking process are shown in Figure 6. The supposed planes are generally estimated from table edges in these scenes. Some supposed planes may not be observed directly because of their small size. Therefore, supposed planes exploit more plane features in the scenes. Besides, adding supposed planes increases the number of plane observation for every frame and add more constraints. Therefore, our proposed method using supposed planes significantly reduces the estimation error, better than PP and PP+S.

Table 2. The number of plane landmarks and the average number of observed planes on ICL-NUIM dataset. $\mathbf{N_l}$ denotes the number of plane landmarks. $\mathbf{N_p}$ denotes the average number of observed planes for every frame. The parameter $\mathbf{N_{sp}}$ denotes the average number of supposed planes for every frame.

Sequence	$\mathbf{N_l}$	$\mathbf{N_p}$	$\mathbf{N_{sp}}$
living_room_0	17	2.67197	0.082152
living_room_1	19	2.91511	0.362319
living_room_2	18	3.79115	0.589103
living_room_3	21	2.70991	0.321515
office_room_0	15	2.78515	0.043103
office_room_1	12	2.69151	0.073520
office_room_2	14	2.87855	0.178104
office_room_3	11	2.86301	0.118082


(**a**) living_room_1

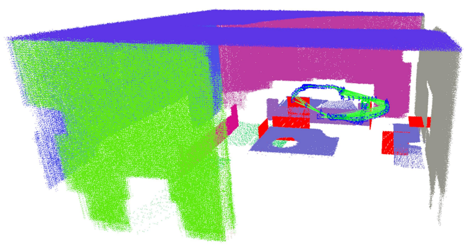
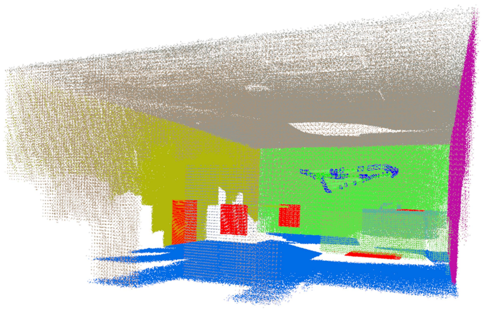
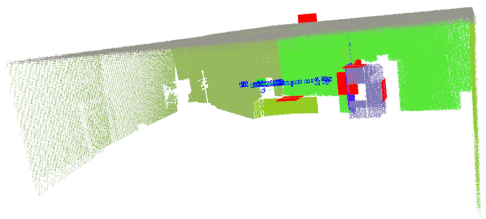

(**b**) living_room_2

Figure 6. *Cont.*



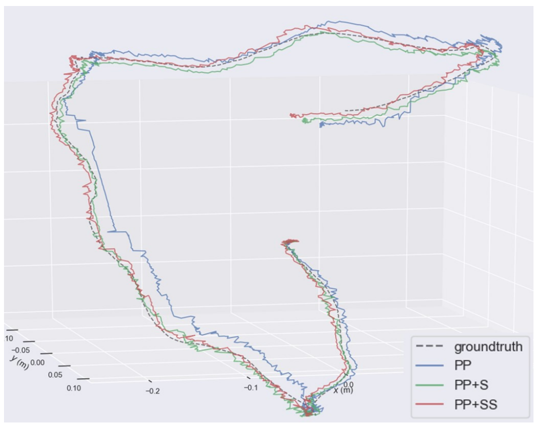
(**c**) office_room_2



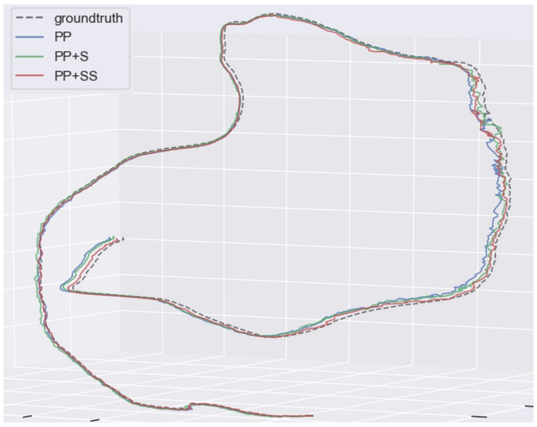
(**d**) office_room_3

Figure 6. Supposed planes generated in the tracking process. Red planes denote supposed planes.

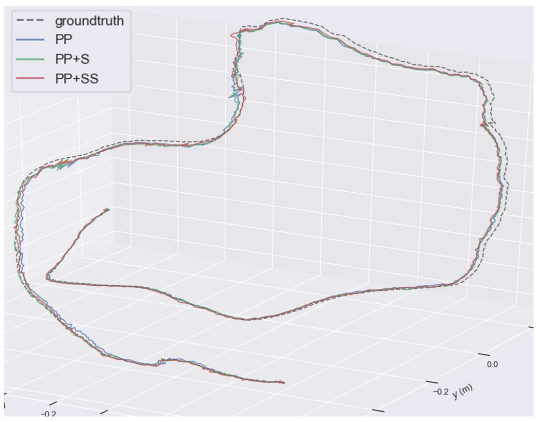
Some comparisons of the estimated trajectories of our proposed methods are shown detailedly in Figure 7. They are all close to the ground truth. But the trajectories of PP have larger drift errors. This also demonstrates that structural constraints help to reduce errors effectively. Supposed planes improve estimation accuracy further.



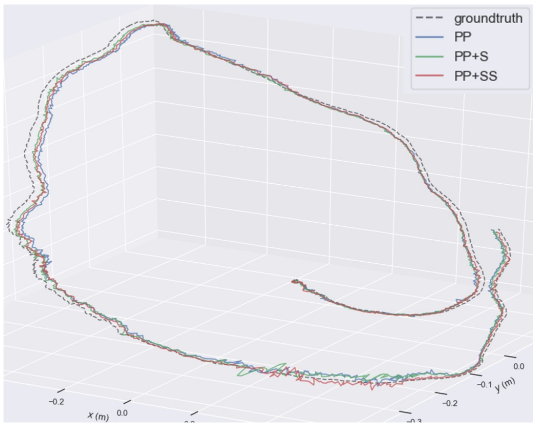
(**a**) living_room_1



(**b**) living_room_2



(**c**) office_room_2



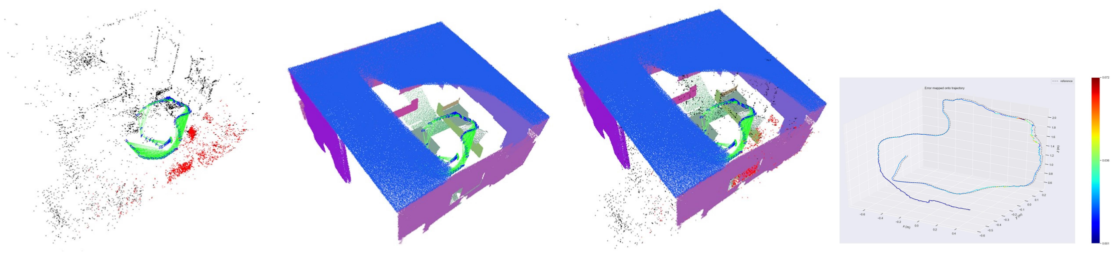
(**d**) office_room_3

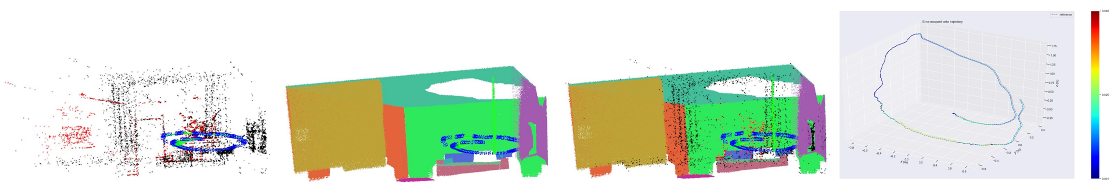
Figure 7. Comparison of estimated trajectories. Different trajectories are plotted together.

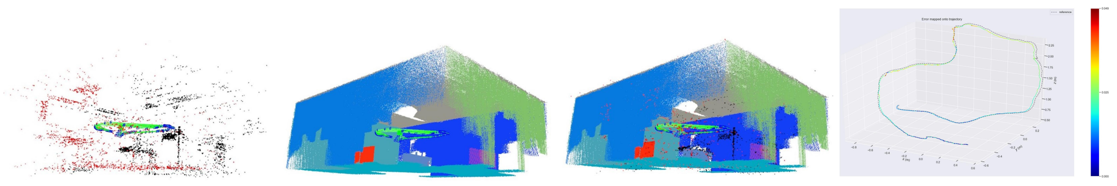
L-SLAM utilizes MW constraints to obtain the drift-free rotation motion of the camera. Unlike our proposed method, they extract the planes parallel with manhattan axis. The 'office_room' sequences are suitable and L-SLAM performs best on these sequences. But our proposed method also works well and is suitable for more scenes, because lots of real scenes can not meet the MW assumption.

We also calculate the weighted average error, and the number of frames serves as the weighting factor. The average error of our method is the smallest. Therefore, our proposed SLAM system improves a lot over the state-of-the-art point-based SLAM system and achieves more robust and accurate estimations in different scenes.

Some trajectories and reconstruction results of our proposed method are shown in Figure 8. Those large planes are extracted and associated accurately. The estimated trajectories are also close to the ground truth.


(**a**) living_room_2


(**b**) living_room_3


(**c**) office_room_2

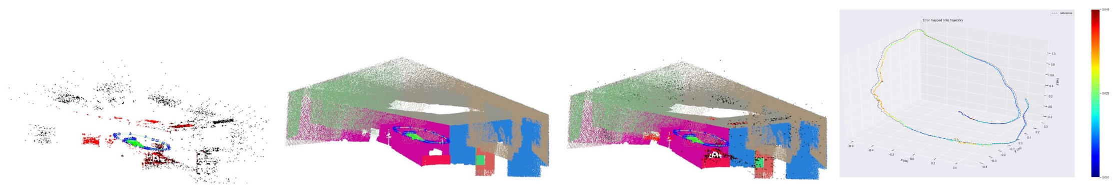

(**d**) office_room_3

Figure 8. Trajectories and reconstruction results. The first column is the point landmarks and the camera trajectories. The second column shows the plane landmarks. The third column shows the plane and point landmarks together. The fourth column is the comparison of the estimated trajectories and corresponding ground truth.

4.2. TUM RGB-D Dataset

We select several sequences containing enough plane features from the TUM RGB-D dataset to evaluate our proposed SLAM system. Because some sequences only have a few features, the original ORB-SLAM2 fails to initialize or track camera poses. Therefore, we modify some thresholds to make ORB-SLAM2 work normally, although the performance may be somewhat worse. We also calculate RMSE to compare the estimation results of different systems, as shown in Table 3. The comparison of the RMSE is also shown in Figure 9. Table 4 shows the average number of observed planes for every frame.

Table 3. Evaluation results of translation ATE RMSE (unit: m) on TUM RGB-D dataset. PP, PP+S, PP+SS denote proposed point-plane SLAM, point-plane SLAM using structural constraints, point-plane SLAM using structural constraints and supposed planes, respectively. Bold numbers represent the best performances.

Sequence	ORB-SLAM2	L-SLAM	PP	PP+S	PP+SS	Frames
fr1/desk	0.020163	-	0.014597	**0.014172**	0.014341	573
fr1/xyz	0.009377	-	0.009433	0.009530	**0.009326**	792
fr3/cabinet	0.075824	0.291	0.043939	0.028968	**0.016801**	1112
fr3/large_cabinet	0.045575	0.14	0.034616	0.032573	**0.029702**	984
fr3/str_notex_far	0.035837	0.141	0.028480	0.024204	**0.022037**	794
fr3/str_notex_near	0.027648	0.066	0.017045	0.015473	**0.012481**	1054
fr3/str_tex_far	0.010693	0.212	0.010438	0.009843	**0.009716**	907
fr3/str_tex_near	0.012139	0.156	0.010039	0.010073	**0.008481**	1057
average	0.031386	0.16927	0.021929	0.018547	**0.015391**	

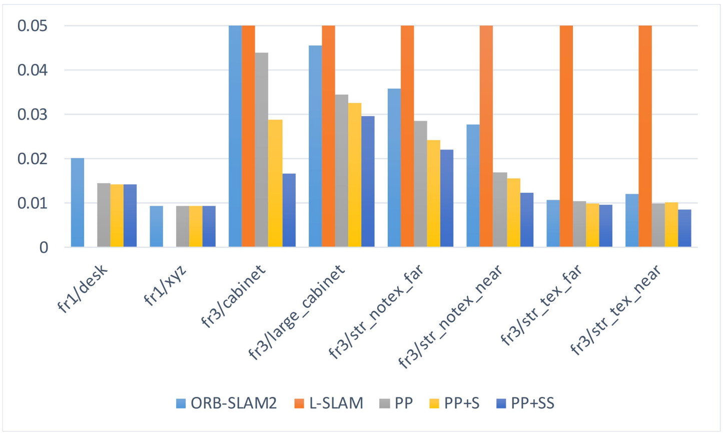


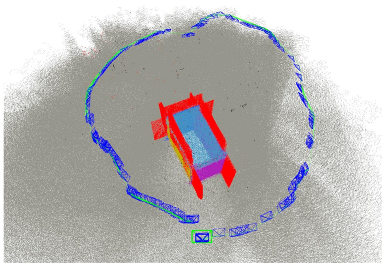
Figure 9. The comparison of the RMSE on TUM RGB-D dataset.

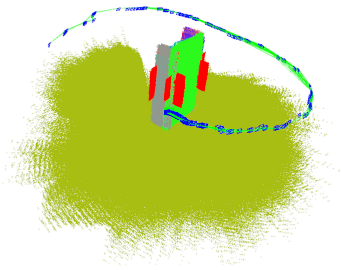
For 'fr1/desk' and 'fr1/xyz', abundant point landmarks ensure the good performance of ORB-SLAM2, while our proposed methods also work well. Because only few planes satisfy the structural constraints, our methods have similar results. For the other sequences, using planes reduces the drift error. It is a challenge for point-based SLAM in low-texture scenes, such as 'fr3/str_notex_far' and 'fr3/str_notex_near'. Besides, there is only one cabinet in the scene in 'fr3/cabinet'. ORB-SLAM2 drifts significantly and even fails to track poses for the entire sequence. But planes provide sufficient constraints. For every frame, the camera can observe three planes at most. Besides, when adding supposed planes, at least one supposed plane can be utilized for every frame, on average. The supposed planes also help to add constraints for other sequences, as shown in Table 4.

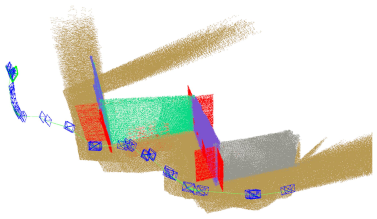
Table 4. The number of plane landmarks and the average number of observed planes on TUM RGB-D dataset. $\mathbf{N_l}$ denotes the number of plane landmarks. $\mathbf{N_p}$ denotes the average number of observed planes for every frame. The parameter $\mathbf{N_{sp}}$ denotes the average number of supposed planes for every frame.

Sequence	$\mathbf{N_l}$	$\mathbf{N_p}$	$\mathbf{N_{sp}}$
fr1/desk	7	1.26876	0
fr1/xyz	4	1.45707	0
fr3/cabinet	6	3.06745	1.281470
fr3/large_cabinet	6	2.04268	0.357724
fr3/str_notex_far	6	2.85264	0.618388
fr3/str_notex_near	5	1.72486	0.299810
fr3/str_tex_far	6	2.94377	0.502756
fr3/str_tex_near	6	2.22422	0.284768

As shown in Figure 10, those red planes are supposed planes generated in the tracking process. These supposed planes are associated with corresponding real planes in the scene, which increase the accuracy of pose estimation. The comparison of the estimated trajectories for some sequences is shown detailedly in Figure 11. Using planes ensures the success of tracking in the entire sequence. Structural constraints reduce the drift error. Besides, the system adding more constraints from supposed planes achieves the best estimation results.


(**a**) fr3/cabinet


(**b**) fr3/large_cabinet


(**c**) fr3/str_notex_far

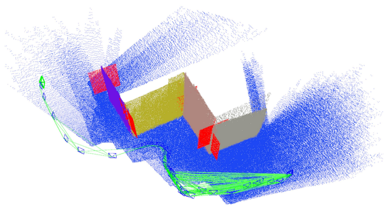
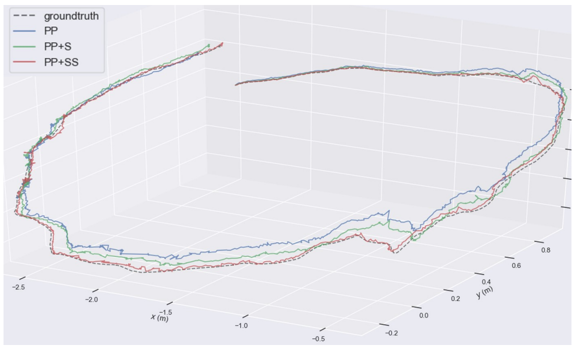
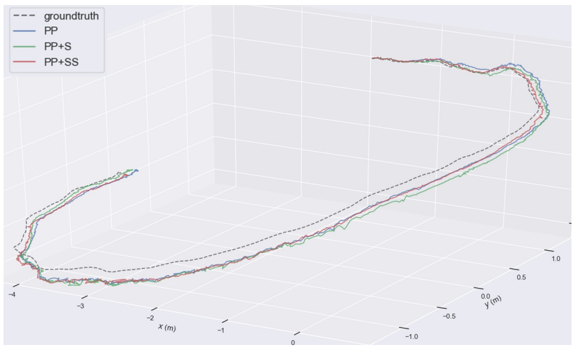

(**d**) fr3/str_tex_near

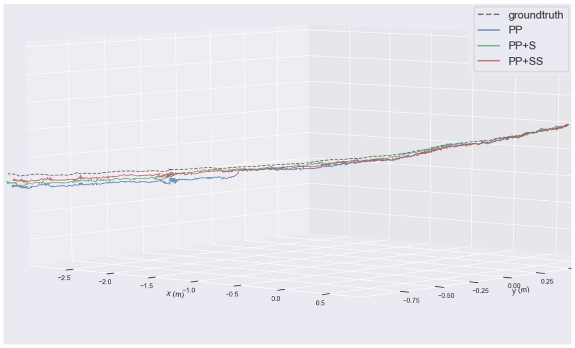
Figure 10. Supposed planes generated in the tracking process. Red planes denote supposed planes.



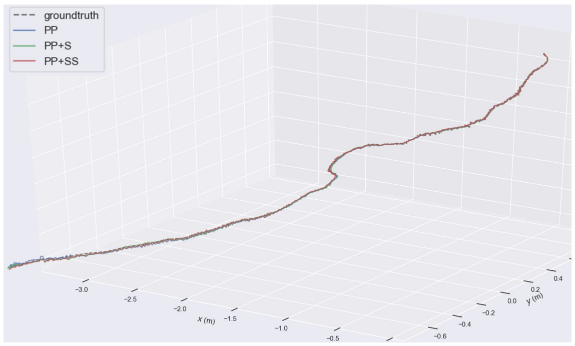
(**a**) fr3/cabinet



(**b**) fr3/large_cabinet



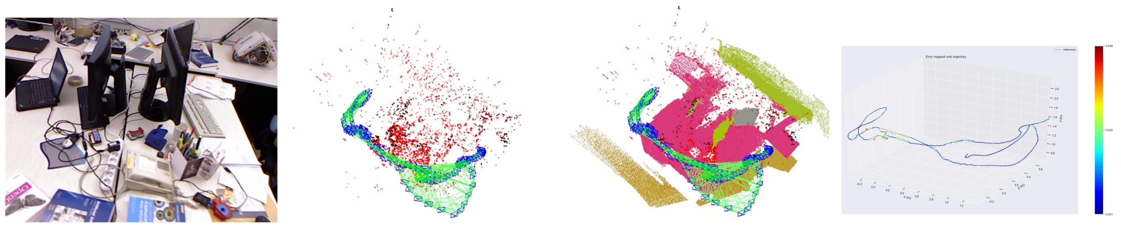
(**c**) fr3/str_notex_far



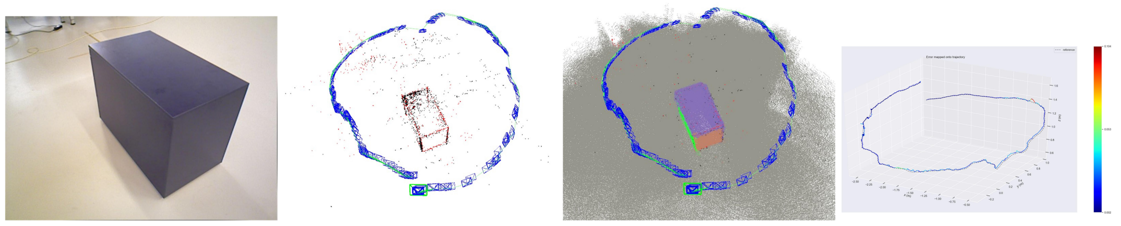
(**d**) fr3/str_tex_near

Figure 11. Comparison of estimated trajectories. Different trajectories are plotted together.

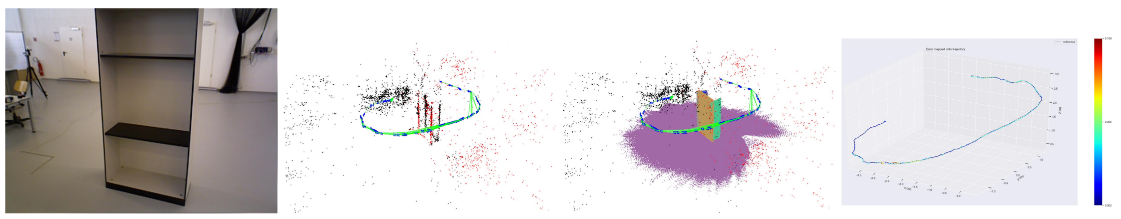
The trajectories and reconstruction results of our proposed method are also shown in Figure 12. It is clear that structural constraints and supposed planes improve the accuracy and robustness of the SLAM system.



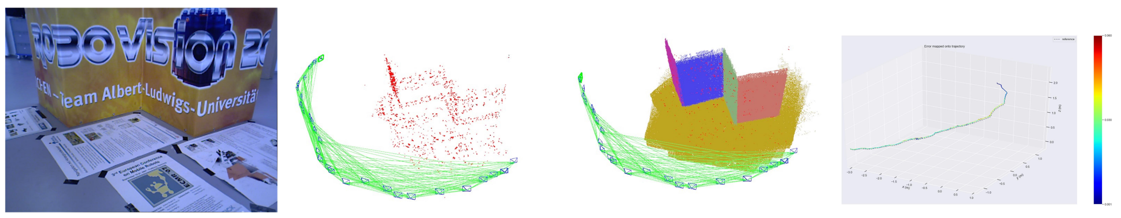
(**a**) fr1/desk



(**b**) fr3/cabinet



(**c**) fr3/large_cabinet



(**d**) fr3/str_tex_far

Figure 12. Trajectories and reconstruction results. The first column is the image showing the scene. The second column shows the point landmarks. The third column shows the plane and point landmarks together. The fourth column is the comparison of the estimated trajectories and corresponding ground truth.

4.3. Runtime Analysis

Table 5 shows the runtime analysis of our system, running on the TUM and ICL-NUIM datasets. All the testing codes are implemented in C++. All the experiments run on a laptop computer with i7-7700HQ 2.80 GHz CPU, 16 GB RAM, without GPU and the operating system is Ubuntu 18.04.

The main difference of our system from other point-based SLAM systems is adding plane features. Therefore, we evaluate the runtime of these additional components. Segmenting planes in organized point clouds is very fast, which takes about 6.8 ms for every frame. Besides, it takes about 7 ms to extract edge lines and generate supposed planes. The runtime of other components is almost the same as point-based SLAM's. Therefore, the additional runtime of the system is about 14 ms, compared with point-based SLAM. It takes about 38 ms to process every frame, which performs nearly in real-time.

Table 5. Average runtime (unit: ms) of different components.

Main Components	**Runtime (ms)**
ORB Extraction	11.54862
Plane Segmentation	6.803255
Supposed Plane Generation	7.060935
Matching and Tracking Landmarks	11.87778
Local Optimization	141.1678
Global Optimization (128 KFs)	344.3516
Frame Tracking	38.07953

5. Conclusions

In this work, we propose a novel point-plane SLAM system for indoor environments. To reduce the drift error, we add structural constraints for those parallel or perpendicular planes. Unlike other plane-based SLAM work, we exploit plane edges to add more reliable constraints. We calculate a supposed perpendicular plane according to the plane and its edge line and treat it like other plane observations. Then the camera poses can be estimated by all of these features. Our proposed algorithm is tested on public benchmarks and achieves more robust and accurate estimation results. Structural constraints can add more constraints between planes, even these planes can not be observed by the same frame. Therefore structural constraints can reduce the drift error for indoor environments. Supposed planes increase the number of plane observations in one frame and add additional sufficient constraints to solve accurate camera poses. Therefore, Our proposed SLAM algorithm is suitable to use in those indoor applications.

Currently, loop detection and relocalization are performed by using feature points. Future work will exploit planes for faster loop detection and relocalization. We will also consider adding constraints between points and planes to remove those bad landmarks.

Author Contributions: Conceptualization, X.Z. and W.W.; methodology, X.Z.; software, X.Z.; validation, X.Z., X.Q. and Z.L.; formal analysis, X.Z.; investigation, X.Z.; resources, W.W.; data curation, X.Z. and Z.L.; writing—original draft preparation, X.Z.; writing—review and editing, W.W. and X.Q.; visualization, Z.L.; supervision, W.W. and R.W.; project administration, X.Z.; funding acquisition, W.W. and R.W.

Funding: This work was supported by the Natural Science Foundation of China under grant number 91748101.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

As we have shown in the main paper, the problem of SLAM can be represented by a factor graph. To estimate the camera poses, we need to solve a nonlinear least-squares problem, such as Equation (13). All iterative methods for nonlinear optimization, whether Levenberg-Marquardt or Gauss-Newton, need Jacobians of the error functions to update the variables.

We use the Lie algebra $\phi \in \mathfrak{so}(3)$ to represent the orientation, and $\boldsymbol{\xi} \in \mathfrak{se}(3)$ to represent the pose during the optimization process. Then the rotation matrice $\mathbf{R} \in \mathbf{SO}(3)$ can be associated by the exponential map $\mathbf{R} = \exp(\boldsymbol{\phi}^{\wedge})$. $(\cdot)^{\wedge}$ is an operator producing an antisymmetric matrix. Similarly, the transformation matrice $\mathbf{T} \in \mathbf{SE}(3)$ can also be associated by the exponential map $\mathbf{T} = \exp(\boldsymbol{\xi}^{\wedge})$.

Using the perturbation scheme [37], the Jocabian of reprojection error $\mathbf{e}_z$ in Equation (8) is given by

$$\frac{\partial \mathbf{e}_z}{\partial \delta \boldsymbol{\xi}} = \frac{\partial \mathbf{e}_z}{\partial \mathbf{P}_c}\frac{\partial \mathbf{P}_c}{\partial \delta \boldsymbol{\xi}} = \frac{\partial \mathbf{e}_z}{\partial \mathbf{P}_c}\frac{\partial \exp(\boldsymbol{\xi}_{cw}^{\wedge})\mathbf{P}_w}{\partial \delta \boldsymbol{\xi}} \tag{A1}$$

$$= -\begin{bmatrix} \frac{f_x}{z_c} & 0 & -\frac{f_x x_c}{z_c^2} \\ 0 & \frac{f_y}{z_c} & -\frac{f_y y_c}{z_c^2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \mathbf{p}_c^{\wedge} \end{bmatrix} \tag{A2}$$

$$= \begin{bmatrix} -\frac{f_x}{z_c} & 0 & \frac{f_x x_c}{z_c^2} & \frac{f_x x_c y_c}{z_c^2} & -f_x - \frac{f_x x_c^2}{z_c^2} & \frac{f_x y_c}{z_c} \\ 0 & -\frac{f_y}{z_c} & \frac{f_y y_c}{z_c^2} & f_y + \frac{f_y y_c^2}{z_c^2} & -\frac{f_y x_c y_c}{z_c^2} & -\frac{f_y x_c}{z_c} \end{bmatrix} \tag{A3}$$

$\mathbf{p}_c(x_c, y_c, z_c)^{\top}$ is the corresponding Euclidean point of $\mathbf{P}_c$.
Similarly, the Jocabian of the error $\mathbf{e}_m$ in Equation (10) is given by

$$\frac{\partial \mathbf{e}_m}{\partial \delta \boldsymbol{\xi}} = \frac{\partial \mathbf{e}_m}{\partial \boldsymbol{\pi}_c}\frac{\partial \boldsymbol{\pi}_c}{\partial \delta \boldsymbol{\xi}} = \frac{\partial \mathbf{e}_m}{\partial \boldsymbol{\pi}_c}\frac{\partial \exp(\boldsymbol{\xi}_{cw}^{\wedge})^{-\top}\boldsymbol{\pi}_w}{\partial \delta \boldsymbol{\xi}} \tag{A4}$$

$$= -\begin{bmatrix} -\frac{n_{cy}}{n_{cx}^2+n_{cy}^2} & \frac{n_{cx}}{n_{cx}^2+n_{cy}^2} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{1-n_{cz}^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0} & -(\boldsymbol{n}_c)^{\wedge} \\ -(\boldsymbol{n}_c)^{\top} & \mathbf{0} \end{bmatrix} \tag{A5}$$

$$= \begin{bmatrix} 0 & 0 & 0 & \frac{n_{cx}n_{cz}}{n_{cx}^2+n_{cy}^2} & \frac{n_{cy}n_{cz}}{n_{cx}^2+n_{cy}^2} & -1 \\ 0 & 0 & 0 & -\frac{n_{cy}}{\sqrt{1-n_{cz}^2}} & \frac{n_{cx}}{\sqrt{1-n_{cz}^2}} & 0 \\ n_{cx} & n_{cy} & n_{cz} & 0 & 0 & 0 \end{bmatrix} \tag{A6}$$

$\boldsymbol{\pi}_c(n_{cx}, n_{cy}, n_{cz}, d_c)^{\top}$ is the plane represented in the camera coordinate system.

The structural constraints only add constraints of orientation. The Jocabians of the error $\mathbf{e}_{sp}$ in Equation (11) and $\mathbf{e}_{so}$ in Equation (12) are the same:

$$\frac{\partial \mathbf{e}_s}{\partial \delta \boldsymbol{\phi}} = \frac{\partial \mathbf{e}_s}{\partial \boldsymbol{n}_c}\frac{\partial \boldsymbol{n}_c}{\partial \delta \boldsymbol{\phi}} = \frac{\partial \mathbf{e}_s}{\partial \boldsymbol{n}_c}\frac{\partial \exp(\boldsymbol{\phi}_{cw}^{\wedge})\boldsymbol{n}_w}{\partial \delta \boldsymbol{\phi}} \tag{A7}$$

$$= -\begin{bmatrix} -\frac{n_{cy}}{n_{cx}^2+n_{cy}^2} & \frac{n_{cx}}{n_{cx}^2+n_{cy}^2} & 0 \\ 0 & 0 & \frac{1}{\sqrt{1-n_{cz}^2}} \end{bmatrix} \begin{bmatrix} -(\boldsymbol{n}_c)^{\wedge} \end{bmatrix} \tag{A8}$$

$$= \begin{bmatrix} \frac{n_{cx}n_{cz}}{n_{cx}^2+n_{cy}^2} & \frac{n_{cy}n_{cz}}{n_{cx}^2+n_{cy}^2} & -1 \\ -\frac{n_{cy}}{\sqrt{1-n_{cz}^2}} & \frac{n_{cx}}{\sqrt{1-n_{cz}^2}} & 0 \end{bmatrix} \tag{A9}$$

References

1. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
2. Li, J.; Zhong, R.; Hu, Q.; Ai, M. Feature-Based Laser Scan Matching and Its Application for Indoor Mapping. *Sensors* **2016**, *16*, 1265. [CrossRef] [PubMed]
3. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 1–10.
4. Wang, R.; Wan, W.; Wang, Y.; Di, K. A New RGB-D SLAM Method with Moving Object Detection for Dynamic Indoor Scenes. *Remote Sens.* **2019**, *11*, 1143. [CrossRef]
5. Meng, X.; Gao, W.; Hu, Z. Dense RGB-D SLAM with Multiple Cameras. *Sensors*. **2018**, *18*, 2118. [CrossRef] [PubMed]
6. Grisetti, G.; Kummerle, R.; Stachniss, C.; Burgard, W. A tutorial on graph-based slam. *IEEE Intell. Transp. Syst. Mag.* **2011**, *2*, 31–43. [CrossRef]

7. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
8. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 834–849.
9. Engel, J.; Koltun, V.; Cremers, D. Direct Sparse Odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 611–625. [CrossRef] [PubMed]
10. Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 15–22.
11. 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 Proceedings of the 211 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Basel, Switzerland, 26–29 October 2011; pp. 127–136.
12. Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. Dtam: Dense tracking and mapping in real-time. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.
13. Weingarten, J.; Siegwart, R. EKF-based 3D SLAM for structured environment reconstruction. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Edmonton, AB, Canada, 2–6 August 2005; pp. 3834–3839.
14. Zureiki, A.; Devy, M. SLAM and data fusion from visual landmarks and 3D planes. *IFAC Proc. Vol.* **2008**, *41*, 14651–14656. [CrossRef]
15. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; The MIT Press: Cambridge, MA, USA, 2006; pp. 330–332.
16. Gostar, A.K.; Fu, C.; Chuah, W.; Hossain, M.I.; Tennakoon, R.; Bab-Hadiashar, A.; Hoseinnezhad, R. State Transition for Statistical SLAM Using Planar Features in 3D Point Clouds. *Sensors* **2019**, *19*, 1614. [CrossRef] [PubMed]
17. Taguchi, Y.; Jian, Y.D.; Ramalingam, S.; Feng, C. Point-plane SLAM for hand-held 3D sensors. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5182–5189.
18. Ma, L.; Kerl, C.; Stückler, J.; Cremers, D. CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 26–21 May 2016; pp. 1285–1291.
19. Kaess, M. Simultaneous localization and mapping with infinite planes. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 25–30 May 2015; Volume 1, p. 2.
20. Ming, H.; Westman, E.; Zhang, G.; Kaess, M. Keyframe-based dense planar SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
21. Ferrer, G. Eigen-Factors: Plane Estimation for Multi-Frame and Time-Continuous Point Cloud Alignment. Available online: http://sites.skoltech.ru/app/data/uploads/sites/50/2019/07/ferrer2019planes.pdf (accessed on 16 August 2019).
22. Yang, S.; Song, Y.; Kaess, M.; Scherer, S. Pop-up slam: Semantic monocular plane slam for low-texture environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016.
23. Coughlan, J.M.; Yuille, A.L. Manhattan World: Compass Direction from a Single Image by Bayesian Inference. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Kerkyra, Greece, 20–25 September 1999.
24. Yi, Z.; Kneip, L.; Rodriguez, C.; Li, H. Divide and Conquer: Efficient Density-Based Tracking of 3D Sensors in Manhattan Worlds. In Proceedings of the Asian Conference on Computer Vision (ACCV), Taipei, Taiwan, 21–23 November 2016.
25. Kim, P.; Coltin, B.; Jin Kim, H. Linear RGB-D SLAM for planar environments. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 333–348.
26. Wang, L.; Wu, Z. RGB-D SLAM with Manhattan Frame Estimation Using Orientation Relevance. *Sensors* **2019**, *19*, 1050. [CrossRef] [PubMed]
27. Guo, R.; Peng, K.; Fan, W.; Zhai, Y.; Liu, Y. RGB-D SLAM Using Point–Plane Constraints for Indoor Environments. *Sensors* **2019**, *19*, 2721. [CrossRef]

28. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G.R. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
29. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2003; pp. 65–68.
30. Trevor, A.J.B.; Gedikli, S.; Rusu, R.B.; Christensen, H.I. Efficient organized point cloud segmentation with connected components. In Proceedings of the 3rd Workshop on Semantic Perception Mapping and Exploration (SPME), Karlsruhe, Germany, 5 May 2013.
31. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
32. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 573–580.
33. Frank, D.; Michael, K. Factor Graphs for Robot Perception. *Found. Trends Robot.* **2017**, *6*, 1–139.
34. Grisetti, G.; Kümmerle, R.; Strasdat, H.; Konolige, K. g2o: A general Framework for (Hyper) Graph Optimization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 3607–3613.
35. Gálvez-López, D.; Tardos, J. D. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197. [CrossRef]
36. Handa, A.; Whelan, T.; Mcdonald, J.; Davison, A.J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–5 June 2014; pp. 1524–1531.
37. Barfoot, T. *State Estimation for Robotics*; Cambridge University Press: Cambridge, UK, 2017; pp. 238–245.