A Submap Joining Algorithm for 3D Reconstruction using an RGB-D Camera based on Point and Plane Features

Jun Wang[a], Jingwei Song[a], Liang Zhao[a], Shoudong Huang[a], Rong Xiong[b]

[a]*Centre for Autonomous Systems, University of Technology, Sydney, Australia*
[b]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, 310027, China*

Abstract

In standard point-based methods, the depth measurements of the point features suffer from noise, which will lead to incorrect global structure of the environment. This paper presents a submap joining based SLAM with an RGB-D camera by introducing planes as well as points as features. There are two steps of this method: submap building and submap joining. Several adjacent keyframes, with the corresponding small patches, visual feature points, and planes observed from these keyframes, are used to build a submap. We fuse the submaps into a global map in a sequential fashion, such that, the global structure is recovered gradually through plane feature associations and optimization. We also show that the proposed algorithm can handle plane association problem incrementally in submap level, as the plane covariance can be obtained in each submap. The use of submap significantly reduces the computational cost during the optimization process, without losing any information about planes. The proposed method is validated using both publicly available RGB-D benchmarks and datasets collected by ourselves. The algorithm can produce good quality trajectories and 3D models on these challenging datasets, which are difficult for existing RGB-D SLAM or SFM algorithms.

Keywords: RGB-D, SLAM, 3D reconstruction, Indoor mapping

1. Introduction

The RGB-D camera is a promising tool for 3D reconstruction in indoor environment. This consumer-level camera combines a depth camera and an RGB camera, i.e., for every frame, there is an RGB image and a depth image. Compared to laser scanners, the camera is cheaper, more flexible and can be easily deployed in indoor environments.

The RGB-D camera has been utilized in robotics for Simultaneously Localization and Mapping (SLAM) research. But to the best of our knowledge, building a high-quality indoor map in real-time is still challenging. The standard method of RGB-D SLAM is the point-based method [19, 6]. The point features are extracted from RGB images using visual feature extractor and matched between different frames using corresponding feature descriptors. The difference with monocular SLAM is that the depth image provides a depth measurement of the visual point feature. This measurement can solve the scale ambiguity problem in pure image-based 3D reconstruction. As a benefit of the depth measurements, the RGB-D 3D reconstruction optimization algorithms can easily converge. However, the depth measurement provided by the depth image is very noisy, the batch optimization algorithm will be easily trapped to a local minimum. For this reason, obtaining a consistent global structure over a long sequence of RGB-D images is nearly impossible by directly applying the traditional methods.

What's more, point-based SLAM has poor performance under texture-less areas. In fact, these areas are mostly planes, which provide useful information of the structure. There are many researches utilizing planes in the literature. CPA-SLAM [17] modeled the environment with a global model consisting of planes, which reduces the drift along with direct image alignment. In their optimization, two types of residuals are defined, one is the distance between points, and the other is the distance from points to the global plane when the points are detected on a plane. This kind of residuals may suffer from the noise of depth images. Kintinuous [31] and Elastic fusion [32] did not used planes explicitly. Instead, they used a frame-to-model strategy to overcome drift. Though providing large-scale dense mapping, these two methods still have drift when recovering global structures because the plane structure is not utilized in the global model, as shown in the results from Section 5. Dense planar SLAM [27] used planes as a representation of the model; however, the planes are not included in the state vector for optimization.

Planes are also explored for different applications and with various sensors. Plane-primitives were applied to provide robust odometry for RGB-D cameras [24]. A novel closed-form algorithm was presented in [23] to estimate the motion and piecewise-planar reconstruction in a stereo rig. Recent research used planes to improve the accuracy of monocular SLAM. Pop-up SLAM [35] used deep convolutional networks to detect planes in images and demonstrated that this kind of structures could enhance both state estimation and dense mapping, especially in texture-less environments. In [9], a structured learning algorithm was proposed to fit the reconstructed model to the "box" structure of the room following the perspective cues. Parallel and orthogonal walls were applied to improve the odometry of the camera [28]. In monocular scenery, the plane structure of the environment is difficult to detect, and these ap-

proaches can only be used in limited scenes under some restrictive conditions.

Besides SLAM, there are also offline reconstruction methods that use planes as extra information in their formulation. In [7], the authors proposed a fine-to-coarse global registration method. In particular, the plane association windows increase gradually during iterations and cover the whole sequence at the end of the algorithm, which is so-called hierarchy optimization. As all the cost functions are computed in each step, a huge amount of computation resources and memory are required in this method. Their experiments report that a 128G RAM is used, and many computations are required for a typical sequence of RGB-D images. In [33], they made use of the labeled objects in the environment to guide the optimization and thus obtain a better global registration. In this method, using the limit extent of one object as constraint can significantly reduce drift and improve the global registration. However labeling objects manually for video sequences is exhausting and costly. Different scenes should be labeled separately and the labels cannot be reused in other future scenes.

Though aforementioned methods utilize planes in their algorithm, the problems are not solved. As there is no appropriate plane descriptor, like Scale-Invariant Feature Transform (SIFT) [16] for points, plane data association remains a difficult topic. Our method associates planes between submaps as the algorithm can provide plane parameter covariance in each submap. Submap based approaches have been used in building large scale map in feature-based SLAM [20, 12, 36]. Tectonic SAM algorithm [20] proposed an efficient submap based approach, in which a divide-and-conquer way was used to fuse the submap into the global map. Sparse Local Submap Joining Filter (SLSJF) [12] presented a canonical and efficient submap joining algorithm that makes use of consistent local submaps to build large-scale feature-based maps. They explored the sparse structure together with a novel state vector and applied a covariance submatrix recovery technique. In [36], they proved that the submap joining problem could be formulated as a linear least squares problem, and thus can be solved with a closed-form solution. All these methods used points as features.

In this paper, we propose a submap joining based 3D reconstruction algorithm using both points and planes as features, which is more efficient than batch offline 3D reconstruction and more accurate than existing RGB-D SLAM algorithms. We compared our method with the state-of-art approaches using several publicly available challenging datasets and also data collected from our robot platform, which demonstrate the improvements of our method in terms of the quality of both trajectory and 3D reconstruction model.

This paper builds on the previous preliminary work [30] with the following major improvements: (1) A novel and effective plane parameterization on manifold is proposed in our methods, which is illustrated in Section 2; (2) A precise submap generating method is proposed using points, patches and planes. Patches and planes are more resistant to the depth measurements noise, and thus can produce a more accurate submap than that from only points. This is described in Section 3; (3) A novel submap joining method using both points and planes as features are proposed. The new way dealing with planes is proved to be much more efficient than the hierarchical style plane utilization [7]. Meanwhile, our formulation can also produce a fine-to-coarse effect on the final registration result. Section 4 presents the details of this method; (4) An improved algorithm and the evaluation on more challenging datasets.

The paper is organized as follows: The proposed method is described in Section 2, 3 and 4. Specifically, this comprises the plane parametrization in Section 2, local map building in Section 3 and local map joining in Section 4. Section 5 presents experiments and results. Section 6 concludes the paper with future directions.

2. A Novel Plane Parametrization

A standard representation of an infinite plane is a four-element vector $\pi = [\pi_1, \pi_2, \pi_3, \pi_4]$, where $\pi_n = [\pi_1, \pi_2, \pi_3]$ is the normal vector of the plane, $\pi_d = \pi_4$ is the distance between the plane and the origin of the coordinate system. As a plane has three DOF (Degree Of Freedom), the above parameterization is overparameterized.

Overparametrization makes the information matrix rank-deficient, which will lead to a singularity problem in Gauss-Newton (GN) optimization. An additional constraint is usually added to this parameterization:

$$\pi_1^2 + \pi_2^2 + \pi_3^2 = 1 \tag{1}$$

which makes the normal vector unit.

Note that the unit normal vector can be presented by an azimuth angle ϕ and an altitude angle θ in the spherical coordinate system:

$$\begin{aligned} \pi_1 &= \sin(\phi)\cos(\theta) \\ \pi_2 &= \sin(\theta) \\ \pi_3 &= \cos(\phi)\cos(\theta) \end{aligned} \tag{2}$$

This formulation turns the four-parameter representation of a plane into three-parameter one, which is a minimal representation, and is preferred in most of the situations. However, we find that this minimal representation makes the convergence significantly slow and not converge to the global minimum even in the simplest case. The following simulation will analyze the phenomenon. $\mathfrak{N}$ random points p_i ($i = 1, 2, ..., \mathfrak{N}$) on a plane π are generated, and we define an objective function as:

$$\mathbb{E} = \sum_{i=1}^{\mathfrak{N}} \|(p_i^x, p_i^y, p_i^z)(\pi_1, \pi_2, \pi_3)^T + \pi_4\|^2 \tag{3}$$

where $p_i = [p_i^x, p_i^y, p_i^z]$ is the i-th point. π_4 is in Euclidean space and our interest in this paper lies on the first three parameters $[\pi_1, \pi_2, \pi_3]$ of the plane. Without losing generality and for easier illustration, we fix the parameter π_4 to be a constant all the time. Equation (3) can then be written as

$$\begin{aligned} \mathbb{E}(\theta, \phi) = \sum_{i=1}^{\mathfrak{N}} \|(p_i^x, p_i^y, p_i^z)(\sin(\phi)\cos(\theta), \\ \sin(\theta), \cos(\phi)\sin(\theta)^T + \pi_4^c\|^2 \end{aligned} \tag{4}$$

where π_4^c is a constant number (π_4^c is set to 1 in the simulation). The ground truth of θ, ϕ are set to θ^g, ϕ^g. As shown in Figure 1, the range of θ and ϕ are set to $[\theta^g - \pi/4, \theta^g + \pi/4]$ and $[\phi^g - \pi/4, \phi^g + \pi/4]$. The aim of the optimization algorithm is to find a minimum objective function value under such parameterization. However, the shape of the objective function is close to an inverted mountain range, and there are so many local minima that the optimization algorithm will not find the global minimum easily. We apply GN algorithm and present the result in Table 1. The algorithm always converges to some local minimum if some small noise is added to the initial value. The results reported in Table 1 agrees with the objective function surfaces shown in Figure 1(a) and (b). To solve the problem, we proposed a novel plane parameterization method based on the manifold. The key idea is that we still use four parameters $[\pi_1, \pi_2, \pi_3, \pi_4]$ to present a plane π, but the plane is updated using three parameters, which is essentially a minimal representation. For simplicity, the fourth parameter π_4 is left out in later discussion. The following equations give the details about the proposed parameterization. Let the two-elements update vector be $u \in \mathbb{R}^2$, the symbol superscribe $^\vee$ be a skew-symmetric mapping as

$$m^\vee = \begin{bmatrix} 0 & -m_3 & m_2 \\ m_3 & 0 & -m_1 \\ -m_2 & m_1 & 0 \end{bmatrix} \tag{5}$$

In iterated optimization methods, such as GN or LevenbergMarquardt (LM), the update vector is added to the currently estimated state vector. In traditional Euclidean space, the update vector is added by a standard addition "+". In our case, the parameters are defined on the manifold, and a special plus operator $\oplus$ is defined for the optimization algorithm.

$$u \oplus {}^i\pi_n = exp((g(u)u)^\vee)({}^i\pi_n) \tag{6}$$

where ${}^i\pi_n$ is the normal part of the reference plane, u is the small update vector, ${}^j\pi_n$ is the updated normal, function $g(u)$ maps update value $u \in \mathbb{R}^2$ to $g(u) \in \mathbb{R}^{3\times2}$. Note that $g(u)$ is not unique and we proposed one $g(u)$ as the null space of the small normal vector:

$$g(u) = \mathcal{N}((sin(u_1)cos(u_2), sin(u_2), cos(u_1)sin(u_2))^T) \tag{7}$$

where $\mathcal{N}$ is the null space function. As $rank(u) = 1$, its null space will be $\mathcal{N} \in \mathbb{R}^{3\times2}$.

As shown in Figure 1(c) and (d), the same dataset is applied using our proposed parametrization. The two horizontal axis are π_2 and π_3, the vertical axis is the total cost function values. Note that as space is limited, variable π_1 is not drawn. There is no ambiguity because our parametrization makes sure that every new estimated vector $[\pi_1, \pi_2, \pi_3]$ is a unit vector. Compared with Figure 1(a) and (b), our objective function surface is more like an inverted mountain peak. Any optimization algorithm will find the global minimum very efficiently.

From Table 1, we can see that the proposed parametrization converges to the global minimum in all of the test cases, the convergence property is much better than the minimal representation, which is easily trapped by the local minimum problem. Moreover, the proposed parametrization is free of any constraints like Equation (1), and thus can be solved using standard GN optimization algorithms.

3. Submap Generating

In the mainstream of our algorithm, there are two distinct steps; one is building submaps based on the keyframes, the other is fusing the submap into global map sequentially. As submap is designed to be built from the local environment, for example on a few keyframes and all the information can be processed in real time, the submap should be as precise as possible. As shown in Figure 2, three kinds of information are extracted from the input images: feature points, small patches, and planes. In the following sections, we will give a detailed illustration of information extraction, data association, and optimization.

3.1. Point feature extraction

In the field of Computer Vision, various feature extraction methods are proposed, for example, Speeded Up Robust Features (SURF) [2], SIFT [16]. SIFT is famous because it is robust in scale and light invariance for long base-line image pairs. SURF improves the speed of feature extraction process, but it is still challenging to be used it in a real-time application with low computation capacity. [26] proposed Oriented fast and Rotated Brief (ORB) by combining Fast corner detection [25] and Binary Robust Independent Elementary Features (BRIEF) descriptor [3], and achieved a better result regarding efficiency and accuracy. ORB is especially suitable for movable platforms and real-time applications, where computation resource is limited. As we are towards a real-time algorithm, ORB is applied to detect visual features in RGB images. Figure 3(a) shows one input RGB image. ORB visual features are detected and highlighted with the green crosses in Figure 3(c).

3.2. Patch extraction

In submap generating, small patches are utilized to contribute to registration and combat noises in depth images. A patch is a small plane structure in a 3D point cloud. As the patch is fitted from a set of points, it is more accurate and robust than single depth measurement. As shown in Figure 3(d), patches are extracted from the depth images. Algorithm 1 presents the patch extraction algorithm, the input of the algorithm is a depth image, the output is a set of small patches. First, we divide the depth image into $s_{grid} \times s_{grid}$ grid cell uniformly. Then RANdom SAmple Consensus (RANSAC) fitting algorithm is applied to extract the plane coefficients for each grid cell separately with a threshold $\mathbb{T}'_\tau$ controlling the distance and $\mathbb{T}_\tau$ controlling the minimal number of inliers. If the patch is confirmed, an anchor point and plane coefficients are selected to identify a patch. The center point of the small patch is regarded as the anchor. As shown in Figure 3, the patches are extracted from the input depth image.

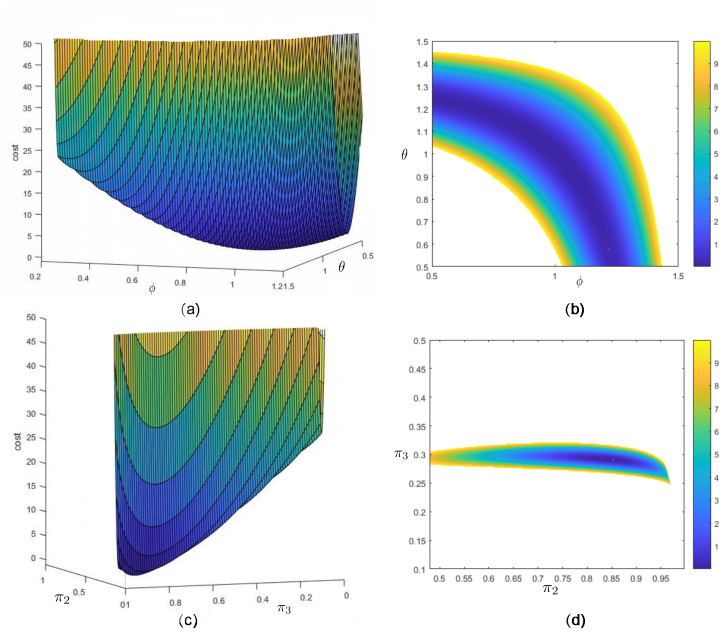


Figure 1: Objective function surface. The objective function values are evaluated on a various combinations of parameters. (a) Objective function from the minimal parametrization of a plane. (b) Project the objective values of (a) to 2D. (c) Objective function from the proposed parameterization of a plane. (d) Project the objective values of (c) to 2D

Parametrization	Initial value noise	Residual error ($\approx$)	Is convergence	Iterations ($\approx$)
Minimal	$\pi/8$	e-3	Yes	15
Proposed	$\pi/8$	e-20	Yes	4
Minimal	$\pi/4$	e-2	Yes	18
Proposed	$\pi/4$	e-20	Yes	5

Table 1: Convergence comparison of the two parameterizations of planes in five runs

3.3. Plane extraction

Though there are several plane extraction methods in the literature [10, 34, 21], we proposed a novel hierarchical clustering method to achieve this job. There are two reasons for this: one is extracting multiple planes from noisy depth image (or point cloud) is non-trivial and cost a lot of computation resources, the other is we have already extracted middle-level patches and a plane can be regarded as a set of patches that lies on the same plane. The proposed hierarchical clustering method explores the distance relationship between the patches. Patches on the same plane will be clustered into one class based on the hierarchical tree structure.

Distance matrix computing. Compared with extracting planes on the dense point clouds, clustering on sparse patches requires less computation, because there are only a few hundreds of patches, but 640×480 points in one depth image. What's more, patches already contains both location and orientation information about the point cloud. In most of the plane segmentation algorithms, there should be an external normal computation step before any further processing. The input of the proposed plane extraction algorithm (Algorithm 2) is a set of patches. The outputs are multiple planes. Firstly, a distance matrix is computed between each pair of the patches. Let $\tau_a^i \in \mathbb{R}^3$ and $\tau_a^j \in \mathbb{R}^3$ be the anchor points, $\tau_n^i \in \mathbb{R}^3$ and $\tau_n^j \in \mathbb{R}^3$ be the normals of i-th and j-th patches. A distance $d_{i,j}^{\tau}$ between two patches is defined as:

$$d_{i,j}^{\tau} = max((\tau_a^i - \tau_a^j)^T \tau_n^i, (\tau_a^i - \tau_a^j)^T \tau_n^j) \tag{8}$$

where $(\tau_a^i - \tau_a^j)^T \tau_n^i$ is actually a projection distance between the difference of anchor points and one patch normal. If τ^i and τ^j share the same plane, the distance will be equal to zero. Note that though there are a few hundreds of patches in one frame, not all of them share the same plane. When computing the distance, we first compare the fourth parameter of the two patches' coefficients, if the difference exceeds a threshold the distance is set to a large value (larger than threshold $\mathbb{T}_{\tau}''$) and can avoid most of the computations of Equation (8). The fourth parameter τ_d of the patch's coefficients is the distance between the patch and the origin of the coordinate system in the local frame.

Hierarchical tree construction. The clustering tree is constructed hierarchically. In the first level, each patch is assigned to its own cluster, which means each cluster contains only one patch. Then find the closest pair of clusters and merge them into a single cluster. We use a maximum patch distance as the

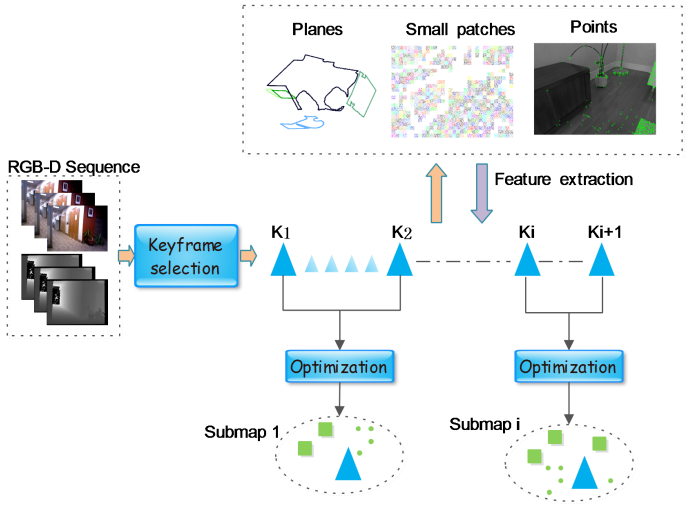


Figure 2: An overview of how submaps are built. The input of submap is several adjacent keyframes with points, patches, and planes. An optimization algorithm is applied to this information based on the energy functions defined in Section 3.6.

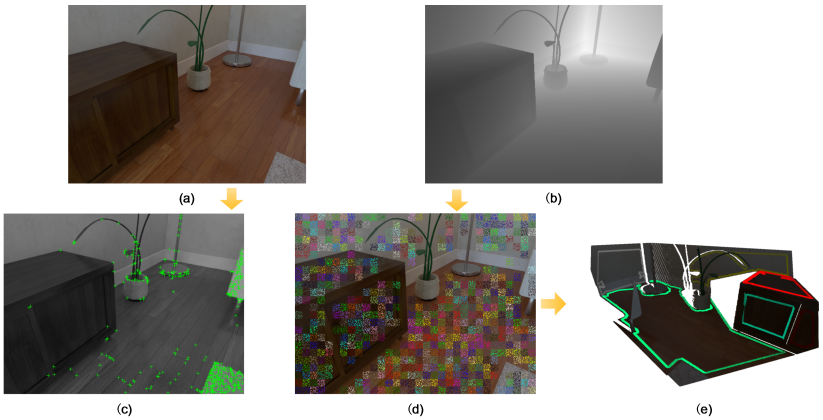


Figure 3: Feature extraction from RGB and depth image. (a) RGB image. (b) Depth image. (c) Feature points are extracted from the RGB image. (d) Patches are detected in each grid cell from the depth image. (e) The patches are then hierarchically clustered to generate large planes.

Algorithm 1: Patch extraction from the depth image

Input: A depth image $\mathcal{I}_D$
Output: A set of patches ζ
Divide the depth image into $s_{grid} \times s_{grid}$ grid cells uniformly.
for i=1;$i \leq s_{grid} \times s_{grid}$;$i++$ **do**
  Project the pixels in the grid cell to 3D points set Γ_p
  for $j = 1; j \leq n_{ransac}; j++$ **do**
    Pick up three points in points set Γ_p: p_1, p_2, p_3
    Generate two vectors from the points $v_1 = [p_1 - p_2]$, $v_2 = [p_1 - p_3]$
    Compute the null space v_3 for $[v_1^T, v_2]$
    Assign the normal of patch $\tau_n = v_3$
    Compute the forth paramter of patch $\tau_d = |p_1^T \tau_n|$
    while *loop over each point* p_k *in* Γ_p **do**
      Compute $d = |p_k^T \tau_n + \tau_d|$
      if $d < \mathbb{T}'_\tau$ **then**
        $n_{inlier} \leftarrow n_{inlier} + 1$
      end
    end
    if $\frac{n_{inlier}}{s_{grid} \times s_{grid}} > \mathbb{T}_\tau$ **then**
      Calculate the anchor points for the patch
      Refine the patch parameters using all the inlier points in Γ_p
      Add patch to ζ
      Break
    end
  end
end

Algorithm 2: Plane extraction algorithm based on hierarchical clustering

Input: A set of patches ζ_τ in one depth image
Output: A set of planes ζ_π
Compute distance matrix
Set each patch as a cluster
while *number of cluster* $n_{cluster} > 1$ *and the closest distance* $d_{min} < \mathbb{T}_\pi$ **do**
  Merge the two closest clusters
  Update distance matrix
end
foreach *Patches in one cluster* **do**
  Fit a plane using the merge function in Equation (9)
end

cluster distance in computing the distance between clusters. As all the patch distances have been stored in the distance matrix, this process is very fast. Repeat the merging until the cluster distance is larger than threshold $\mathbb{T}''_\tau$. Figure 4 presents how the patches are merged according to the patch-to-patch distance hierarchically.

Each cluster contains the patches from the same planes. GN algorithm is applied to each cluster of patches and compute the coefficients of each plane. The objective function is defined as the sum of distances between patch anchors to the plane. The following equation defines the objective function.

$$E^{\pi}_{cluster} = \sum_{i=1}^{n_\tau} |{\tau^i_a}^T \pi_n + \pi_d|^2_\Sigma \tag{9}$$

where π_n is the normal parameter, π_d is the distance parameter of the plane π, n_τ is the number of the patches in one cluster (one plane), τ^i_a is the anchor point of i-th patch.

3.4. Data association

Data association is a process that identifies the same feature between two or more frames, in our case, it is the process that finds the corresponding visual points, patches, and planes in sequential images.

As the visual point features come with a position and a descriptor, which can be used to match two sets of points. In our experiments, we use point feature descriptors to initialize the matching and apply RANSAC to detect the outliers. With the matched visual point features, a rough relative pose can be computed using 3D feature measurements from both RGB and depth image. Unfortunately, patches and plane features have no feature descriptors, but only geometry features. In our study, the following methods are proposed to help to find the correspondences for patches and planes.

Patch association Suppose patches $\tau^l_i, (i = 1, 2, 3, ...)$ are from l-th frame, $\tau^{l+1}_j, (j = 1, 2, 3, ...)$ are from $(l+1)$-th frame. For comparison under a common coordinates, patches τ^{l+1}_j are transformed to the l-th frame, using the following equation

$${}^l\tau^{l+1}_j = \mathcal{H}(\mathcal{T}_l^{-1}\mathcal{T}_{l+1}, \tau^l_j) \tag{10}$$

where ${}^l\tau^{l+1}_j$ are the transformed patches, $\mathcal{T}_{l+1} \in SE(3)$ and $\mathcal{T}_l \in SE(3)$ are the poses of l-th and $(l+1)$-th frames, function $\mathcal{H}$ transforms the plane parameters from local to global coordinates. Then a KD-tree is constructed using the anchor points from patches τ^l_j. KD-tree builds a spatial index over the 3D anchor points. With the spatial index, we can search the nearest neighbor patches with little computational cost. For every patch in ${}^l\tau^{l+1}_j$, a KD-tree search request is made using the patch anchor, and n'_τ nearest candidate patches returned. Note that n'_τ pair of patches are only close in anchor positions but contain no information about normal directions. Thus there is no guarantee that they are close in patch-to-patch distance as expressed in Equation (8). So the patch distance from Equation (8) is applied on the n'_τ pair of patches. The minimal patch distance is compared with a threshold; if the distance is smaller than the threshold, then the patch pair is regarded as corresponding patches on the two frame. This procedure is similar to the point-to-plane ICP method.

Plane association. Like the first step in patch association, all the planes are transformed into a common coordinate frame. Let π^l_j be the planes in l-th frame, ${}^l\pi^{l+1}_j$ be the transformed planes from $(l+1)$-th frame. For simplicity, we leave out the subscribe and superscribe, and let π' and π'' be the pair of planes to be evaluated.

From Section 3.3, we obtain the plane estimation π by GN algorithm, and every estimation has an information matrix $I_\pi \in$

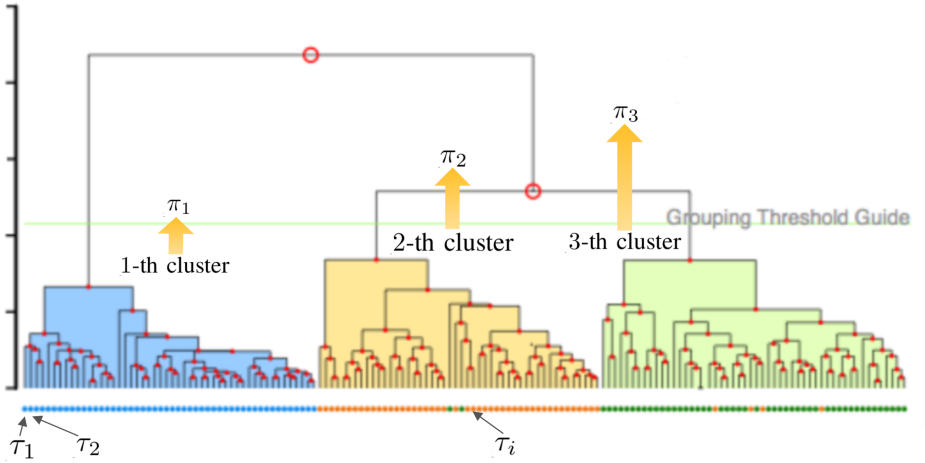


Figure 4: Plane extraction from a set of input patches using hierarchical clustering algorithm.

$\mathbb{R}$. The Mahanolabis distance can be defined as:

$$D_\pi = (\pi' - \pi'')^T (I_{\pi'} + I_{\pi''})(\pi' - \pi'') \tag{11}$$

Let the covariance matrix be Σ_π, which is the inverse of the information matrix. From Cholesky decomposition, we can find a matrix C, which satisfies

$$\Sigma_\pi = CC^T \tag{12}$$

Let

$$y = C^{-1}(\pi' - \pi'') \tag{13}$$

If π' and π'' are the same plane, y will follow a standard Gaussian distribution and the distance in Equation (11) become

$$\begin{aligned} D_\pi &= y^T C^T (CC^T)^{-1} Cy \\ &= y^T y \\ &= \sum_{i=1}^{k} y_i^2 \end{aligned} \tag{14}$$

we can see that the defined distance follows a χ^2 distribution with $k = 4$. Thus a constant threshold is chosen to validate if two planes are the same feature. The threshold is a validation gate, which is a region of acceptance such that $100(1-\ni)\%$ true correspondence are rejected. In our experiments, we choose the confidence level $\ni$ to be 0.95.

3.5. Uncertainty analysis of points, patches and planes

There are two sources of uncertainties, one is the RGB image, the other is depth image. The visual features in RGB image are relatively stable and the variance of them is usually regarded as one pixel. In this study, we pay more attention to the uncertainty from the noisy depth image. [14] calibrated the RGB-D camera and gave an uncertainty model for the depth measurements. For a depth measurement u_z, its variance can be expressed as

$$\sigma_z = \frac{1}{fb}\sigma'_c u_z^2 \tag{15}$$

where f is the focal length of the IR camera, b is the baseline between IR projector and receiver, σ'_c is a parameter related with of RGB-D camera device.

Let a point in 3D be $p = (x, y, z)$ with covariance P_p

$$P_p = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{bmatrix} \tag{16}$$

In the covariance matrix, we already know the element σ_z. The relationship between 3D points in local frame and image pixels can be described as a pinhole model

$$d\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{17}$$

where $d = z$ in RGB-D case, f_x and f_y are the focal length, c_x and c_y are the principle points of the RGB camera. From Equation (17), we have the relationship between x, y, z and u, v. Usually $\sigma_u = 1$, $\sigma_v = 1$, and we suppose that u and v are independent variables. We can define the variance of x and y as follows

$$\begin{aligned} \sigma_x^2 &= \frac{(\sigma_z^2(u - c_x)^2 + \sigma_u^2 z^2)}{f_x^2} \\ \sigma_y^2 &= \frac{\sigma_z^2(v - c_y)^2 + \sigma_v^2 z^2}{f_y^2} \end{aligned} \tag{18}$$

and $\sigma_{xz}, \sigma_{yz}, \sigma_{xy}$ as

$$\begin{aligned} \sigma_{xz} = \sigma_{zx} &= \frac{u - c_x}{f_x}\sigma_z^2 \\ \sigma_{yz} = \sigma_{zy} &= \frac{v - c_y}{f_y}\sigma_z^2 \\ \sigma_{xy} = \sigma_{yx} &= \frac{(u - c_x)(v - c_y)}{f_x f_y} \end{aligned} \tag{19}$$

The generation of patch relies on the 3D points. Suppose patch τ_i is generated from n_p points $[p_1, p_2, ...p_{n_p}]$. The uncertainty

of the center point and the parameters for a patch is computed. The parameters of a patch are solved by GN iteration. So the covariance of the parameters of a patch can be given by

$$P_\tau = (J_\tau^T \Sigma_\tau J_\tau)^{-1} \tag{20}$$

The Jacobian matrix J_τ is evaluated at the last iteration step. For the covariance of center points, the mean of all the covariance of the points is regarded as the covariance of the center point of the patch.

3.6. Optimization for one submap

Data extraction and association for the submap have been illustrated from Section 3.1 to 3.4. In this section, we will describe how to recover the camera poses and feature variables with all these information.

The common representations for camera rotation matrix include rotation matrices, quaternions, angle-axis representation or yaw-pith-roll Euler angles. The rotation has 3 DOF, any representations that use more than three parameters are over-parameterization. According to this, 9-elements rotation matrix and 4-elements quaternions are over-parameterization. The Euler angles are minimal representations and used extensively in the literature. However, there is a typical issue known as "gimble lock". Moreover, in map joining, a wrap on the three angles should be applied in every iteration, as the updated angle may exceed a predefined interval, for example $[-\pi, \pi]$. We apply $SO(3)$ Lie group to represent the rotation matrix to avoid these issues. A $SO(3)$ group is a special orthogonal group that fits the properties of a rotation matrix, and at the same time keep many properties from Lie group. The $SO(3)$ group has an associated $so(3)$ algebra, which is the tangent space around the identity element of the group. In our optimization, the three-element vector $so(3)$ acts as the state vector. The $so(3)$ can be transformed to $SO(3)$ group using a symmetric skew operator followed by an exponential map, which will be defined in the following sections.

3.6.1. Energy function

Energy term on point features. A 3D point feature is measured as a pixel location (u, v) from the RGB image, and a depth value d from the depth image. As there are many outliers in the visual point matches, and the depth value suffers from noise, a robust Huber cost function is applied to filter these outliers. A standard reprojection error between matched 3D points p in world coordinates and measurements is adopted.

$$E_p = \sum_{i=1}^{n_p} \sum_{l} \rho(\|(u_i, v_i, d_i) - \mathcal{P}(\mathcal{R}^l p_i + T^l)\|^2_{\Sigma_p}) \tag{21}$$

where ρ is the robust Huber cost function and Σ_p is the covariance matrix associated with the scale of the visual feature points and depth measurements. In our experiments, u and v are considered as independent and assigned one-pixel variance. The depth value has a unit of meter, so a relatively large weight is assigned to it. Projection function $\mathcal{P}$ takes a 3D local coordinates as input and outputs the projected pixel values, which can be expressed as

$$\mathcal{P}((X, Y, Z)^T) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ Z \end{bmatrix} \tag{22}$$

where f_x, f_y, c_x, c_y are the focal length and principal points of the camera. In our study, these intrinsic parameters are assumed to be known from calibration.

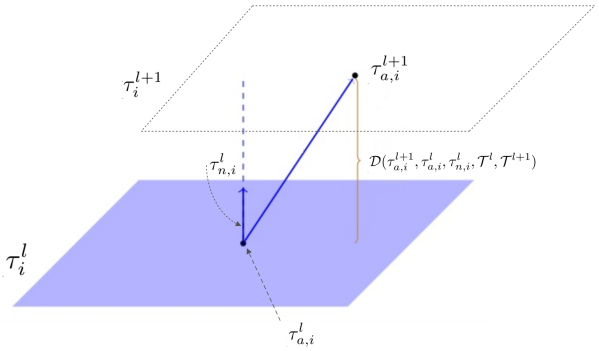


Figure 5: The geospatial relationship between two associated patches τ_i^l, τ_i^{l+1} from the l-th and $(l+1)$-th frame. $\tau_{a,i}^l$ and $\tau_{a,i}^{l+1}$ are the anchor points, $\tau_{n,i}^l$ is the patch normal. Function $\mathcal{D}$ is the distance that will be minimized in optimization as defined in Equation (23)

Energy term on small patches. The basic idea of utilizing these small patches is doing a 3D registration in a patch-based ICP style. A rough pose estimation from points provides initial values for the patch associations. This kind of registration utilizes the small patch coefficients fitted from the depth values, rather than the point clouds transformed from the depth images, the later usually suffer from much noise from the sensor itself.

As shown in Figure 5, two associated patches from two frames should be coplanar in the global frame. The distance function $\mathcal{D}$ will be zero if the poses are adjusted to be perfect. We define an error term as follows to minimize the distance between two corresponding patches, aiming at obtaining coplanar patches.

$$\begin{aligned} E_\tau^{l,l+1} = &\sum_{i=1}^{n_\tau} \rho(\|(\mathcal{R}^l \tau_{a,i}^l + T^l - \mathcal{R}^{l+1} \tau_{a,i}^{l+1} - T^{l+1})^T \mathcal{R}^l \tau_{n,i}^l\|_{\Sigma_\tau}) \\ &+ \sum_{i=1}^{n_\tau} \rho(\|(\mathcal{R}^l \tau_{a,i}^l + T^l - \mathcal{R}^{l+1} \tau_{a,i}^{l+1} - T^{l+1})^T \mathcal{R}^{l+1} \tau_{n,i}^{l+1}\|_{\Sigma_\tau}) \end{aligned} \tag{23}$$

where $\mathcal{R}^l$, $\mathcal{R}^{l+1}$ are the rotation matrix, T^l, T^{l+1} are the translation vectors, $\tau_{a,i}^l$, $\tau_{a,i}^{l+1}$ are the anchor points on i-th corresponding patches, $\tau_{n,i}^l$, $\tau_{n,i}^{l+1}$ are the normal vectors on the I-th corresponding patches of the l-th and $(l+1)$-th frames, n_τ is the number of corresponding patches between the two frames. Briefly, this term measures the patch-to-patch distance between the anchor points along patch normals. During iterations of the optimization algorithm, the correspondences of patches will be updated according to the current estimated pose of the two adjacent frames. Ideally, the distance vector of anchor points should

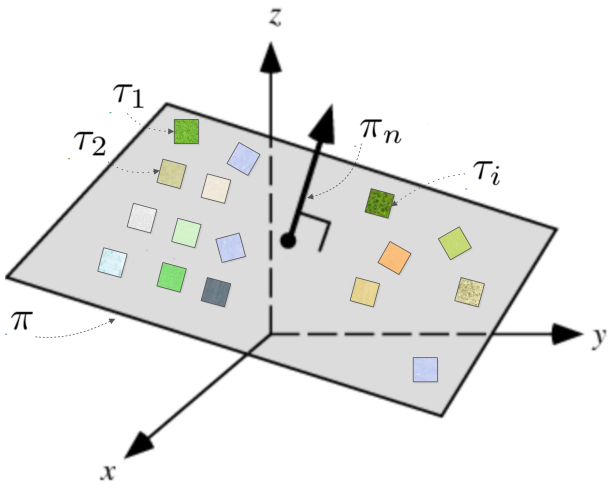


Figure 6: The geospatial relationship between a plane and associated patches $\{\tau_1, \tau_2 ... \tau_i ...\}$ from different frames. Anchor points form these patches should be on the plane π.

be exactly vertical with each normal of the patches, and the residual error will be zero in such case. In practice, the optimization algorithm tries to adjust the camera pose to make the two vectors as vertical as possible. Note that in the cost function in Equation (23), the camera poses are the only variables and the coefficients of the patches are considered as constants.

Energy term on planes with patches. In this part, we describe the relationship between planes and small patches. In a local submap, there are n_k keyframes, and a plane can be observed by n'_k ($n'_k <= n_k$) keyframes. The association relationship has been identified in plane extraction and data association sections (Section 3.3 and 3.4). If a plane π is observed by keyframe $\mathcal{F}$, the plane is then associated with the patches from the cluster of that frame. The plane coefficients are optimized using the fact that patches on different keyframes are on the same plane, as shown in Figure 6. An energy term is defined to measure the relationship between planes and patches:

$$E_\pi = \sum_{l=1}^{n'_k} \sum_{i=1}^{n_\tau} \rho(\|(\mathcal{R}^l \tau_{a,i} + T^l)\pi_n + \pi_d\|^2_{\Sigma_{\tau_i}}) \tag{24}$$

where π_n is the normal part and π_d is the distance part of the plane coefficients. This geometric error between patches and plane is minimized, aiming at obtaining the plane coefficients. As the benefits of the proposed plane parameterization, we do not need to add additional constraints to make the plane representation unique.

Finally, all the constraints within a submap are put together to optimize, and the following objective function is defined

$$E_{M_s} = W_p E_p + W_\tau E_\tau + W_\pi E_\pi \tag{25}$$

where the three terms are from point features, small patches, and planes detected and associated within a submap, the weight parameters W_p, W_τ, W_π will be illustrated in the experiment section. The problem is optimized using GN method. The output of submap is the optimal solution of the state vector and information matrix as defined by

$$(\hat{X}^L, I^L) \tag{26}$$

where $\hat{X}^L$ (the superscript "L" stands for the local submap) is an estimate of the state vector

$$X^L = (X^L_{c1}, X^L_{c2}, ...X^L_{cn_k}, X^L_{p1}, X^L_{p2}, ...X^L_{pn_p}, X^L_{\pi 1}, X^L_{\pi 2}, ...X^L_{\pi n_\pi}) \tag{27}$$

and I^L is the corresponding information matrix, which is computed using the estimated variables in the last iteration. The state vector contains the camera final pose X^L_c (the subscript "c" stands for the camera), point features X^L_p and the plane coefficients X^L_π. Since only a few poses, and the points and planes observed by these poses, are included in a submap, the convergence of the optimization is very fast and stable. This makes the real-time implementation possible. The information from patches and planes contributes to the estimation of camera poses.

3.6.2. *Detailed formulation and solving*

For a more detailed illustration about how to solve the above local submap generating problem, the following factor graph formulation is provided to estimate the random variables of camera poses, points and planes, given the measurements from previous sections. A factor graph is used to present the estimation problem as a graphical model [15]. In our factor graph, variable nodes include poses $\mathcal{T}$, points p, and planes π. Factor nodes relate to point measurements m_p, plane measurements m_π and patch measurements m_τ. As we build every local submap on its first pose frame, a prior $\mathfrak{p}$ is added to the first pose.

The factor graph is actually a bipartite graph $G = (\mathcal{F}, \mathcal{X}, \epsilon)$ with three types of nodes: factor nodes $f_i \in \mathcal{F}$ relate the measurements described in previous sections, variable nodes $x_i \in \mathcal{X}$ represent the variables stored in the state vector, i.e. poses, points and planes. Edge is used to connect variable nodes with a factor node. The factor graph G defines the dependence relationship between factors, variables, and edges. A factorization of function of the graph is defined as

$$f(\mathcal{X}) = \prod_i f_i(\mathcal{X}_i) \tag{28}$$

Each factor relates some of the variables encoded in the edge. The goal of the estimation is to find a $\mathcal{X}$ that maximizes the factorization

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} \prod_i f_i(\mathcal{X}_i) \tag{29}$$

As we assume that the measurement noise follows Gaussian distribution, the factorization is equal to

$$f_i(\mathcal{X}_i) \propto exp(-\frac{1}{2}\|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma) \tag{30}$$

where h is a measurement function for one edge. Combine Equation (29) and (30), we can see that maximize Equation

(29) is equivalent to minimize a negative log function of it, so we have

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_i \|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma \tag{31}$$

which is essentially the case defined in our objective function.

Every edge is provided with the corresponding Jacobian matrix, which is the partial derivatives of error function over the variables. For detailed Jacobian matrix, please refer to Appendix A.

In our submap generating, a few adjacent keyframes are selected to construct a submap. The first pose of $(i+1)$-th submap is the end pose in the i-th submap, i.e., there is one shared pose between two submaps, which is an important design for the following submap joining process. In implementation, the submap generating process is implemented as a stand-alone thread, which is independent with other threads, such as tracking, feature extraction.

3.6.3. Robust kernel function

The least squares estimator is very sensitive to the noise of the data. If one datum is bad, i.e., outlier, the model will also be perturbed to fit this datum. There are many techniques to make the estimator more robust to outliers. In this study, we apply M-estimator to deal with the outliers and choose Huber as the robust kernel.

Let r_i be the residual of the i-th datum of the objective function. The standard least-squares method tries to minimize all the residual errors, that is to say, if the residual of one datum is large, the influence of that datum will also be large on the adjustment of the parameters. If the datum is an outlier, the residual is large, and the whole estimation will be corrupted by the datum.

The Huber robust defines a kernel function as

$$\rho = \begin{cases} x^2/2 & |x| < \mathcal{T}_h \\ k(|x| - k/2) & |x| \geq \mathcal{T}_h \end{cases} \tag{32}$$

where $\mathcal{T}_h$ is a threshold determined by χ^2 methods. The Equation (31) then can be transformed into the following minimizing problem

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_i \rho(h_i(\mathcal{X}_i) - z_i) \tag{33}$$

Solving the above equation equals to solving the following iterated reweighted least-square problem

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \sum_i \omega(h_i(\mathcal{X}_i) - z_i)\|h_i(\mathcal{X}_i) - z_i\|^2 \tag{34}$$

where $\omega(h_i(\mathcal{X}_i) - z_i)$ is a weight that should be updated with each iteration.

As we use the Hessian matrix as information matrix in each processing steps, applying the Huber robust function will destroy the weights of the data source. To solve this problem, the optimization is run a few iterations with robustifier to identify the outliers. Then an optimization without robustifier and outliers is applied again to obtain the final estimates.

4. Local Submap Joining

The input of this stage is the local submaps built using the approach described in Section 3. The local submap joining algorithm merges all the local submaps sequentially. The output is a global map, containing all the poses, the points, the coefficients of the planes, and the corresponding information matrix. The major difference between this algorithm and the SLSJF [12] is that we explore the global structure by associating the planes in an incremental fashion.

In batch Bundle Adjustment (BA) with a long sequence of RGB-D images, the algorithm is hard to converge to a high-quality solution. One reason is that the depth image is very noisy, as the sensor is a consumer level RGB-D camera. Moreover, wrong data associations of points, patches, and planes also make it hard to converge to the right solution. In our study, the motivation is trying to apply plane clues to guide the algorithm to converge to a good solution. Figure 7 intuitively shows how this idea works.

The reason why we cannot use plane constraints directly in batch BA is the camera pose solution drift a lot over the long sequence of the scan; it is nearly impossible to identify a single threshold to associate these planes on different frames under such drift. Figure 7 gives an overview of our submap joining algorithm. The submaps are generated using the algorithm described in Section 3. The map joining algorithm will join the submaps one by one, and at the same time, associating the planes in an incremental way. As the drift between two submap is small, it is easier to find a plane correspondence in our method. The associated planes guide the global map to converge to a better solution, as shown in the bottom row of Figure 7. All the submaps will be joined into a single global map in the last map joining step.

4.1. The state vector in global map

The global map starts from the first local submap and expands with the fusing of them. After the fusion of 1 to $j-1$ local submaps, the global map can be denoted by $(\hat{X}^G_{j-1}, I^G_{j-1})$, where $\hat{X}^G_{j-1}$ is an estimate of state vector X^G_{j-1}, I^G_{j-1} is the corresponding information matrix, and the superscript "G" stands for the global map. The following equation gives the detail of the state vector X^G_{j-1}

$$X^G_{j-1} = (X^G_{c_1}, X^G_{c_2}, ...X^G_{c_{n_c}}, X^G_{p_1}, X^G_{p_2}, ...X^G_{p_{n_p}}, X^G_{\pi_1}, X^G_{\pi_2}, ...X^G_{\pi_{n_\pi}}) \tag{35}$$

where X^G_c are the robot poses, X^G_p are the points and X^G_π are the plane coefficients. When fusing a local submap into the global map, the new poses, points, planes in the local submap are added to the global map as new variables, and the first pose of the submap, points and planes associated with existing ones in global map will also be incorporated. The global map will cover all the features at the end of the fusion stage. As shown in Figure 8, X^G_{j-1} is the global map from the last fusion, X^L_j is the current local submap, X^G_j is the new global map, which is the fusion result of X^L_j and X^G_{j-1}. The colored strip denotes the variable nodes in the map. The green nodes are the new

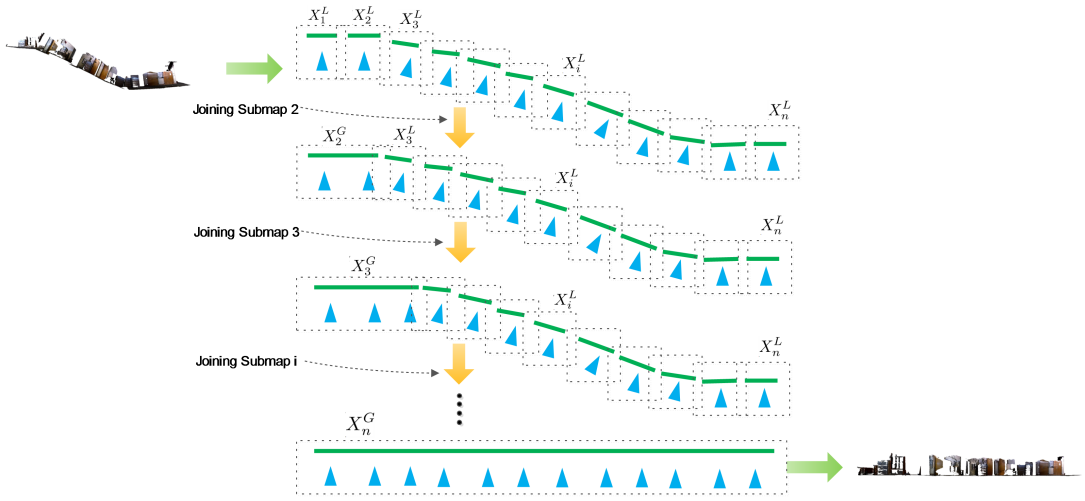


Figure 7: Overview of the submap joining process. The green lines indicate the planes, and the small triangles represent camera poses. The submap joining procedures help the optimization to converge to a reasonable solution with the help of plane features. X_i^L denotes the i-th local submap and X_j^G denotes the j-th global map.

nodes which never exist in the global map, so these nodes are appended to the new global map as new variable nodes.

4.2. Fusion as a least squares problem

We formulate the fusion of local submap with global map as a least squares problem, which takes two part of measurements as input: global map from the previous step and current local submap to be fused.

For local submap $(\hat{X}_j^L, I_j^L)$, we believe that the estimation of the poses and features are locally accurate and can be regarded as a measurement of the true values, with Gaussian noise. The covariance matrix can be given by the information matrix.

$$\hat{X}_j^L = H_j(X_j^G) + w_j \tag{36}$$

where H_j is a transformation function, which transforms the variables from global frame to current local frame using the start pose of the local submap, which is the last pose of global map, w_j is the zero-mean Gaussian "observation noise", whose covariance matrix is

$$P_j^L = (I_j^L)^{-1} \tag{37}$$

Similarly, the global map $(\hat{X}_{j-1}^G, I_{j-1}^G)$ can also be regarded as measurements of the true values of poses and plane coefficients, with Gaussian noise. The covariance matrix can also be given by the information matrix, that is

$$\hat{X}_{j-1}^G = X_{j-1}^G + W_{j-1} \tag{38}$$

where W_{j-1} is the zero-mean Gaussian "observation noise", whose covariance matrix is $P_{j-1}^G = (I_{j-1}^G)^{-1}$.

From the above, the fusion of the j-th local submap to the $(j-1)$-th global map can be formulated as a least squares problem, using all the information from the global and the local submap. The equation below shows this weighted least squares problem

$$\begin{aligned} \underset{X_j^G}{\operatorname{argmin}} \; & (\hat{X}_j^L \ominus H_j(X_j^G))^T I_j^L (\hat{X}_j^L \ominus H_j(X_j^G)) \\ & + (\hat{X}_{j-1}^G \ominus \mathcal{A}_j X_j^G)^T I_{j-1}^G (\hat{X}_{j-1}^G \ominus \mathcal{A}_j X_j^G) \end{aligned} \tag{39}$$

where $\mathcal{A}_j$ is a matrix that extract corresponding variables from X_j^G, which already exist in $\hat{X}_{j-1}^G$. In Section 4.3, we will define the different types of $\ominus$ according to variable nodes in the fusion.

4.3. Definitions of measurement model

The measurements involved in map fusion can be roughly classified into two types: one is the measurement from global map X_{j-1}^G, and the other is the measurement from current local submap X_j^L. Different factor for these measurements and corresponding variables nodes are defined in following equations.

Let $f_{\mathcal{R}}^G$ be the factor between camera pose node in the new global map X_j^G and camera pose measurement in the previous global map X_{j-1}^G. Note that the translation part T of the camera pose is just a 3-element vector in Euclidean space, and can be defined as a normal minus operator, which is similar to the points factor in Equation (42) and left out here.

$$\begin{aligned} f_{\mathcal{R}}^G &= \mathcal{R} \ominus \hat{\mathcal{R}} \\ &= log(\mathcal{R}^T \hat{\mathcal{R}})^\wedge \end{aligned} \tag{40}$$

where the superscript $\wedge$ means the inverse skew-symmetric operator, transforming the matrix in Lie Algebra $so(3)$ to $SO(3)$ Lie Group space.

Let $f_{\mathcal{R}}^L$ be the factor between camera pose node in the new global map X_j^G and camera pose measurement in the current local submap X_j^L.

$$\begin{aligned} f_{\mathcal{R}}^L &= \mathcal{R} \ominus (\mathcal{R}^e \hat{\mathcal{R}}^L) \\ &= log(\mathcal{R}^T \mathcal{R}^e \hat{\mathcal{R}}^L)^\wedge \end{aligned} \tag{41}$$

One variable node $\mathcal{R}$ in this factor is the camera pose associated with the measurement from local submap, the other variable node $\mathcal{R}^e$ is the camera pose associated with the end camera pose from previous global map X_{j-1}^G.

The factor connecting a point node and the measurements from global map X^G_{j-1} can be defined as

$$\begin{aligned} f^G_p &= p \ominus \hat{p} \\ &= p - \hat{p} \end{aligned} \tag{42}$$

The factor connecting a point node and the measurements from local submap X^L_j can be defined as

$$\begin{aligned} f^L_p &= p \ominus (\mathcal{R}^e \hat{p}^L + T^e) \\ &= p - \mathcal{R}^e \hat{p}^L - T^e \end{aligned} \tag{43}$$

In the following part, a measurement model for the plane coefficients (normal part) will be given. Let matrix A be the result of function $g(u)$ in Equation (7), and it is composed of two column vectors α_1, α_2

$$A = g(u) = [\alpha_1, \alpha_2] \tag{44}$$

From the update equation for the plane normal in Equation (6), we have

$$exp((Au)^\vee)^i\pi_n = {}^j\pi_n \tag{45}$$

A solution of Au in Equation (45) is

$$Au = \arccos({}^i\pi_n{}^T {}^j\pi_n)({}^i\pi_n \times {}^j\pi_n) \tag{46}$$

Let L_{Au} to be the left side of Equation (46), R_{Au} be the right side.

$$\begin{aligned} L_{Au} = Au &= [\alpha_1, \alpha_2] \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ &= \alpha_1 u_1 + \alpha_2 u_2 \end{aligned} \tag{47}$$

$$R_{Au} = \delta\pi_n = \arccos({}^i\pi_n{}^T {}^j\pi_n)({}^i\pi_n \times {}^j\pi_n) \tag{48}$$

From Equation (47) and Equation (48), we have

$$\alpha_1 u_1 + \alpha_2 u_2 = \delta\pi_n \tag{49}$$

We can see from Equation (7), matrix A is the null space of vector u, thus the column vectors α_1 and α_2 are the unit orthogonal vectors, i.e.

$$\begin{aligned} |\alpha_1| = |\alpha_2| &= 1 \\ \alpha_1^T \alpha_2 &= 0 \end{aligned} \tag{50}$$

Multiply α_1^T to the both sides of Equation (49), and get

$$u_1 = \alpha_1^T \delta\pi_n \tag{51}$$

Also multiply α_2^T to both sides of Equation (49)

$$u_2 = \alpha_2^T \delta\pi_n \tag{52}$$

Put the above two equations together; we can obtain the factor f^G_π connecting the plane node and measurements from global map X^G_{j-1}

$$\begin{aligned} f^G_\pi = {}^j\pi_n \ominus {}^i\pi_n &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ &= A^T \arccos({}^i\pi_n{}^T {}^j\pi_n)({}^i\pi_n \times {}^j\pi_n) \end{aligned} \tag{53}$$

The factor f^L_π connecting the plane node and measurements from local submap measurements can be defined as

$$\begin{aligned} f^L_\pi &= {}^j\pi_n \ominus \mathcal{R}^{ei}\pi_n \\ &= A^T \arccos({}^i\pi_n{}^T \mathcal{R}^{ej}\pi_n)({}^i\pi_n \times \mathcal{R}^{ej}\pi_n) \end{aligned} \tag{54}$$

The distance part π_d of plane coefficients is in Euclidean space and can be defined similarly with points definition as Equation (42) and left out here.

4.3.1. Definitions of jacobian matrix

To solve the map joining algorithm efficiently, Jacobian matrix of every edge in the factor graph should be defined. The derivatives to points and planes are similar to those described in Section 3.6, so we only give the detailed derivatives on rotation part of camera poses.

Let e^G_R be the corresponding edge of factor f^G_R. From Equation (40) and update equation for $SO(3)$ we have

$$\begin{aligned} f^G_R &= log(R^T \hat{R})^\wedge = -log(\hat{R}^T R)^\wedge \\ &= -log(\hat{R}^T exp(\epsilon^\vee) R_{op})^\wedge \end{aligned} \tag{55}$$

where ϵ is the incremental vector for the rotation matrix, R_{op} is rotation matrix from last state (or iteration), $exp(\epsilon^\vee)R_{op}$ is the update equation for $SO(3)$ Lie Group.

In Lie group, the adjoint function performs the transformation from a tangent vector from the tangent space around one element to the tangent space of another [1]. Equation (55) can be transformed as

$$\begin{aligned} f^G_R &= -log(exp(Adj(\hat{R}^T)\epsilon)^\vee)\hat{R}^T R_{op}) \\ &= -log(exp((\hat{R}^T\epsilon)^\vee)\hat{R}^T R_{op})^\wedge \end{aligned} \tag{56}$$

where Adj is the adjacent function for $SO(3)$. According to BCH (Baker-Campbell-Hausdorff) formula [1], we can get the BCH approximations for f^G_R using only the left Jacobian

$$f^G_R = -(J_l(\psi_2)^{-1}\psi_1 + \psi_2) \tag{57}$$

where

$$\begin{aligned} \psi_1 &= \hat{R}^T \epsilon \\ \psi_2 &= log(\hat{R}^T R_{op})^\wedge \end{aligned} \tag{58}$$

As ϵ is the small incremental vector for the update, ψ_1 should also be a small vector. The derivative over R then can be defined as

$$\frac{\partial e^G_R}{\partial R} = -J_l(\psi_2)^{-1}\hat{R}^T \tag{59}$$

The edge e^L_R of factor f^L_R can be derived similarly by applying BCH on $SO(3)$ and left out here. The Jacobian matrix is composed of these partial derivatives of all the edges in the factor graph.

4.4. Data association

Data association in map joining algorithm is a process of tracking features in a current local submap with those in global map of the last step. Data association is a crucial step as false positive associations make wrong observations, and add false factor nodes in the graph, which can not be detected and marked as outliers easily in further optimizations. In our methods, we propose an appropriate data association for point and plane features in map joining. An overview of the data association in map joining is shown in Figure 8. Details about point and plane features association will be illustrated in detail in the following part.

Algorithm 3: Find point feature correspondence between global map and local submap in map joining

Input: A set of point features in local submap ζ_p^L ; A set of point features in global map ζ_p^G; The end pose $\mathcal{R}_e$ in global map X_{j-1}^G

Output: A point correspondences matrix C_p^{LG} from local and global map; A set of new points from local submap $\zeta_{p'}^L$

Transform points ζ_p^G from global to local frame using $\mathcal{R}_e$, resulting in points set ${}^L\zeta_p^G$

foreach F_i *in local submap* X_j^L **do**
  Compute a frustum for F_i
end

Filter out points that don't lie in any frustums of X_j^L. Add the remaining points to $\bar{\zeta}_p^G$

Project $\bar{\zeta}_p^G$ to each F_i image space

foreach p_i^L *in* X_j^L **do**
  Choose n'_p points from $\bar{\zeta}_p^G$ in image space
  Compute Hamming distance between p_i^L and n'_p points
  Assign the minimal distance to $\mathfrak{D}_{min}$, and the second minimal distance to $\mathfrak{D}_{sec}$
  if $\mathfrak{D}_{min} < \alpha\mathfrak{D}_{sec}$ **then**
    Add p_i^L and the corresponding point with minimal distance to C_p^{LG}
  else
    Add p_i^L to $\zeta_{p'}^L$
  end
end

In every keyframe F_i, a viewing frustum is generated using pinhole model. As the depth camera can only work in a limited depth range $[L_{min}, L_{max}]$, the view frustum is bounded by this extent. In point feature association, every point in current local map tries to find a correspondence in global map. The number of point features in global map is substantial, so it is not wise to compare each pair of point features. We can safely assume that most of the correspondence is inside frustums of keyframes in the local submap. We first transform the global points to local frame and filter out points that do not lie in any frustums of the local map, as described in Algorithm 3. Then visual feature descriptors are used to do the point feature matching, which is a standard solution for visual feature tracking.

The data association described in Algorithm 4 is applied to find the plane correspondences between local and global map. Let π^G be a plane feature in the global map, P^G be the corresponding covariance matrix. Let π^L be the plane feature in local submap, P^L be the corresponding covariance matrix. The plane feature in global map π^G is transformed to local frame of the current local submap $\bar{\pi}^G$ using

$$\bar{\pi}^G = H(\pi^G, \mathcal{T}_e) \tag{60}$$

where H is a transformation function, $\mathcal{T}_e$ ("e" stands for end pose in global map) is the end pose of global map X_{j-1}^G. Function H transforms plane coefficients π^G from the global frame to the local frame using the end pose $\mathcal{T}_e^G$. In transformation, the normal part π_n and distance part π_d of a plane are transformed separately.

$$H(\pi^G, \mathcal{T}_e) = ((\mathcal{R}_e)^{-1}\pi_n^G, T_e n^G + \pi_d^G) \tag{61}$$

where $\pi^G = \{\pi_n^G, \pi_d^G\}, \mathcal{T}_e^G = \{\mathcal{R}_e, T_e\}$. The Mahalanobis distance is defined to measure the distance from the two planes in the same coordinates.

$$\mathcal{D}_\pi^{LG} = (\bar{\pi}^G \ominus \pi^L)^T (P^G + P^L)^{-1} (\bar{\pi}^G \ominus \pi^L) \tag{62}$$

Thus a proper threshold value $\mathbb{T}_\pi$ can be selected based on χ^2 distribution, such that the null hypothesis that the two features are the same one is not rejected under some confidence level. This procedure is repeated at every iteration.

5. Experiments and Evaluation

In this section, we evaluate our proposed algorithm using a series of experiments designed to test the performance of both camera pose estimation and 3D reconstruction. Comparisons are made with state-of-art methods. The following part describes the datasets, the performance comparisons, quantitative evaluations of the camera pose trajectory and qualitative evaluations of the reconstructed scenes. The proposed 3D reconstruction algorithm is implemented using C/C++ under the framework of ROS (Robot Operating System) [22] in Linux, and all the experiments are executed on a laptop with an Intel Core i5-6300, 2.30GHz and 8G RAM. There are three parallel threads in the system: tracking using point features, local submap building, and loop closure. The parameters are set as follows: $W_p = 1, W_\tau = 1, W_\pi = 2, s_{grid} = 20(pixels)$.

5.1. Datasets and evaluation metric

Three publicly available datasets are used in the assessments: ICL-NUIM synthetic scenes [8], Princeton University SUN3D dataset [33] and TUM datasets [29]. TUM dataset is a large RGB-D dataset containing RGB-D videos and ground-truth data with the goal to establish a benchmark for the evaluation of SLAM and SFM systems. The RGB-D video contains the color and depth images from a Microsoft Kinect sensor with the

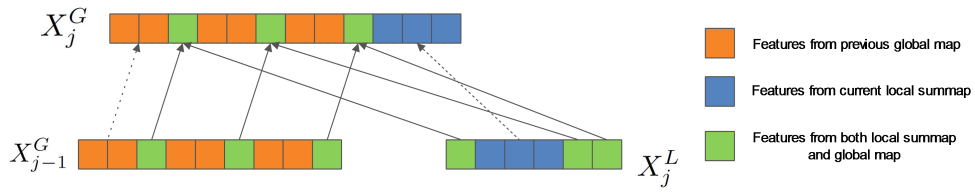


Figure 8: Data associations in submap joining process. The features in the new global map X_j^G come from three sources: features from only previous global map X_{j-1}^G; features from only current local submap X_j^L; features from both X_{j-1}^G and X_j^L

Algorithm 4: Find plane correspondence between global map and local submap in map joining

Input: A set of planes in local submap ζ_π^L; A set of planes in global map ζ_π^G; The end pose $\mathcal{R}_e$ in global map X_{j-1}^G

Output: A plane correspondences matrix C_π^{LG} between local and global map; The set of new planes from local submap $\zeta_{\pi'}^L$

foreach π_i^G *in* ζ_π^G **do**
  Transform π_i^G from global to local frame by applying Equation (60)
  Add transformed plane ${}^L\pi_i^G$ to set ${}^L\zeta_\pi^G$
end
foreach π_i^L *in* ζ_π^L **do**
  Compute Mahalanobis distances between π_i^L and ${}^L\pi_j^G$ in ${}^L\zeta_\pi^G$
  Assign the minimal distance to $\mathcal{D}_{min}^\pi$
  if $\mathcal{D}_{min}^\pi < \mathbb{T}_\pi$ **then**
    Add π_i^L and the corresponding plane with $\mathcal{D}_{min}$ to C_π^{LG}
  else
    Add π_i^L to $\zeta_{\pi'}^L$
  end
end

ground-truth trajectory of the sensor. The videos were recorded at full frame rate (30HZ). The ground truth poses of the sensor were obtained from a high-accuracy (relative to the sensor measurement accuracy) motion-capture system with eight high-speed tracking cameras with a higher frame rate (100HZ).

The ICL-NUIM dataset contains synthetic scenes of a living room and an office room. In the creation of the dataset, the Kintinuous algorithm [31] is first used to scan the room and build a 3D surface of the scene, and then synthetic camera poses are simulated in the 3D model using ray casting. Some artificial noise is added to the observations aiming at evaluating the robustness of SLAM and 3D reconstruction algorithms.

Another popular public RGB-D dataset is SUN3D [33]. It contains a set of 415 RGB-D videos captured with an ASUS Xtion PRO LIVE sensor in a hand-held way. The greatest feature of this dataset is that it include large varieties of challenging scenes. One video usually covers the whole level or multi rooms, which is quite difficult for most of the 3D reconstruction algorithms. As the operators move quite fast with random paces, many images in the dataset are blurred (or skewed) and the depth images are noisy. Though ground truth camera poses are not available in the SUN3D dataset, it provides the distinct, complete geometric structure that can be used to compare the reconstructed surfaces in qualitative comparisons.

To validate our methods further, we collect a dataset using a Kinect sensor mounted on our robot platform. The details of the scene will be described in Section 5.2.

The evaluation metric is borrowed from [29], which is based on a modified version of Hon's approach [11]. First, the correspondence of camera poses is determined using the timestamp in both trajectories. Second, a rotation and translation matrix is computed through singular value decomposition. The rotation and translation matrix is then applied on every poses in the estimated trajectory. The transformed poses are then compared with those of the ground truth trajectory in terms of translation error. An RMSE (Root-Mean-Square Error) is then computed using the following equation

$$e_{rmse} = \sqrt{\sum_i^{n_c} (e_i^T e_i)/n_c} \tag{63}$$

where e_i is error of the estimated i-th pose, n_c is the number of camera poses.

5.2. Methods compared

The proposed methods are compared with other state-of-art RGB-D SLAM [13, 5, 18, 31, 4, 27] and SFM methods [33]. These methods are chosen because: 1) The methods utilize various information, such as direct tracking on all pixels, virtual scans from the model, planes as landmarks. 2) Both online and offline methods are included. As our work target a near real-time algorithm, it is necessary to compare with other online methods. At the same time, we use extra planes as constraints aiming to get a high-quality 3D model. The performance can be illustrated with comparisons to offline SFM algorithms.

Quantitative Comparisons In this section, we compare the proposed methods with alternative methods on datasets from TUM and ICL-NUIM, as they provide the ground truth camera pose trajectory. Trajectory accuracy is an important aspect of SLAM and SFM algorithm, the environment (represented by features) are linked to the trajectory. If the trajectory is not accurate, the map will also be incorrect. All the methods are evaluated on different datasets using the recommended parameters or parameters from the published papers.

Table 2 shows the comparison result. In the video sequence *lrkt*0, there are many planes (walls) in the scenes, and our

method benefits most. The extracted patches can refine the local map building, and the planes can guide the algorithm converges. However in the sequence of $fr3/office$, the scene was collected in clustered environments, and there are rarely planes.

Our methods explore the plane constraints in an elaborate way, and utilize patches refining the relative poses, based on the initial result of point features. The camera poses are improved compared with other alternative methods.

Qualitative Comparisons To evaluate the 3D reconstruction performance of our approach we compared the global map with the output of other state-of-art methods. As shown in Figure 9a, in the output of batch BA of [33], which does not use any plane information, there is a gap between two walls. We evaluated our approach on the same dataset [8], it can be seen from Figure 9b that the global structure is well reconstructed. Figure 9c and Figure 9d show more details of the two models, demonstrating the remarkable improvements of our method. It is quite straightforward to see that the plane constraints help a lot in the refinement of the details. The ceiling will not be fitted together (as one plane) if the constraints of plane are not incorporated in our algorithm. Generally speaking, the walls around the room, the floor and the ceiling provide important clues for the 3D reconstruction, but many methods ignored this information or the planes are not made full use.

We also compared with some state-of-art real-time RGB-D SLAM. It is a little unfair as the real-time application have some trade-off between accuracy and efficiency. However our method also runs in a near real-time fashion, the local map building process is a parallel thread with the tracking process. In sceneries like this, the local map building may delay with half minute on our laptop. Figure 10a shows the result of Kintinuous [31] on the dataset from SUN3D [33], which was captured along a long corridor. Kintinuous failed on this challenging scene because of tracking errors. Elastic Fusion [32] performs well on most of the dataset, but also drifts near the end of the corridor, as shown in Figure 10b. Figure 10c shows the output of our methods. In this case, the patches and planes from the floor in the scene guide our optimization to converge to a solution that fits the planes from different frames of the RGB-D video.

We also validated the proposed methods on our robot platform. A Kinect is mounted on the robot and it traversed through two rounds on our Engineering Building located at the University of Technology, Sydney. A total of 1670 RGB-D image pairs are logged along with the corridor. No other sensor data are logged in our experiments. The corridor is about 25m long. As Figure 11 shows the reconstructed surface is fitted well to the long corridor.

Figure 12 shows a series of experiments on typical scenes from the SUN3D dataset. Theses scenes cover a single room or multi rooms. The planes (walls) in these scenes gives the overall structure for the environment. The result from [33] shows that it is not easy to recover the structure of the scene if only point features and point matches are utilized, as the point feature measurements from depth image are noisy and the uncertainty is too hard to deal with. They achieve good local alignments but in some cases, it fails to recover the structure. It takes hours for [33] to process a single sequence of the video, as it compares each pair of images to find more links between frames. Our method takes about ten minutes to process one sequence of the dataset.

The rightmost column of Figure 12 shows the models of our 3D reconstruction results. Our method overcome the large-scale drift over long sequence, as shown in Figure 12(a). From the view of "point features", these scans indeed cover long sequences, and the algorithm will drift as the scene become larger and larger. However, for the plane features, we can always observe some common feature from different frames, for example, one wall is observed by many scans in each dataset. That's why our methods can deal with the drift but [33] suffer from it.

The middle column of Figure 12 shows the detail comparison of the two methods. The patches in adjacent keyframes refine the results from point features. As an added benefit, the patches are less noisy; they can help to identify points with large uncertainty using robust kernels. These points can be deleted or assigned to large variance values.

5.3. Efficiency analysis of the proposed method

In traditional batch BA, all the information are incorporated (computed) at each iteration, which is very costly for large dataset. Our proposed method improved this by reducing the number of terms in the cost function significantly.

As shown in Table 3, the number of terms in objective functions are calculated both in traditional batch BA and the proposed method. The objective functions are separated by terms related to poses, points, patches and planes, and a total number of the terms are given in the rightmost column.

In Table 3, the parameter n_c is the number of cameras, n_p is the number of points, $n_{p'}$ is the mean observations of each point, $n_{\tau'}$ is the mean number of corresponding patches between two camera poses, n_π is the number of planes and $n_{\tau''}$ is the mean number of patches on each plane. In most of our experiments, $n_{\tau'}$ is around 100, $n_{\tau''}$ is around 40.

In traditional BA, the number of pose terms equals to zero, as the number is counted on the points term, and one point is observed by one pose at a time. The patches term are involved over the whole sequence of the poses, which is the most computational extensive part. Moreover, the patch correspondence relationship between each adjacent poses should be updated after each iteration as the relative poses may be adjusted during state vector update.

However, in our proposed submap joining based methods, these patches are only included in the submap generating, where only several poses exist. The patches are not regarded as variables in state vector, and they will not appear in the submap joining process. Note that this does not mean the information about patches are missing during submap joining, on the contrary, the information about patches is included in the information matrix in the output of a submap. The information matrix will be later incorporated into global map in the joining process.

Similarly, the relationship between planes and patches also requires a lot of computation resources. In BA, every plane should be registered with the associated patches. In the proposed method, this information is also incorporated but only in

Algorithms Datasets	DVO SLAM [13]	RGB-D SLAM [5]	MRS Map [18]	Kintinuous [31]	RGBDTAM [4]	Dense Planar SLAM [27]	Ours
lr kt0	0.104	0.026	0.204	0.072	N/A	0.246	0.019
fr1/desk	0.021	0.026	0.043	0.037	0.023	N/A	0.024
fr3/office	0.035	0.032	0.042	0.030	0.027	N/A	0.028

Table 2: The RMSE of the absolute trajectory error (m) of our algorithm in comparison to previous methods on ICL-NUIM and TUM datasets.

Methods	Poses	Points	Patches	Planes	In total
Traditional BA	0	$3n_p n_{p'}$	$2n_c n_{\tau'}$	$n_\pi * n_{\tau''}$	$3n_p n_{p'} + 2n_c n_{\tau'} + n_\pi n_{\tau''}$
Proposed	$6n_c$	$3n_p$	0	$4n_\pi$	$6n_c + 3n_p + 4n_\pi$

Table 3: Comparison of number of terms in objective functions in batch BA and our map joining method

the submap generating procedure. In submap joining, the plane comes with four variables and associated information matrix, which contains the information about the relationship between patches. In total, the proposed methods can achieve 2 to 5 HZ processing rate.

6. Conclusion and Future Work

In this paper, we present a submap based 3D reconstruction algorithm using points and planes as features. We demonstrate that visual point features suffer from the noise of the depth image, however, the plane features are more stable. The proposed algorithm takes advantage of planes in structured indoor scenes, thus can be used to build a high-quality 3D model. The algorithm is very efficient as it applied submap joining technique, which restricts the most computation extensive part within each local submap. We formulate a novel and effective plane parameterization on manifold, which shows better convergence properties in the experiments. In submap generating, the algorithm utilizes points, patches, and planes. Patches and planes are more resistant to the depth measurements noise and can produce a more accurate submap than using only points. The global structure is gradually recovered by joining new submaps, which introduces a novel way to deal with plane associations and shows to be more efficient than the hierarchical style plane utilization. The experiments demonstrate that the 3D models produced by our algorithm are much better than the state-of-art RGB-D SLAM algorithms and SFM algorithms, and our algorithm is more efficient than offline SFM reconstruction.

In the future, we will improve the algorithm so that it can robustly handle large scenarios with dynamic objects. We will also extend the work to environments with non-planar regions and common objects, such as chairs and desks. The active perception for robust RGB-D SLAM in dynamic environments is also our future research topic.

Acknowledgement

This research is supported in part by SKLICT scholarship and CSC scholarship (grant number: 201604910831).

Appendix A. Jacobian matrix in submap generating

In this part, we will derive the different types of Jacobian matrix involved in submap generating.

For an edge $e_p^{\mathcal{T}}$ connecting point feature p and pose $\mathcal{T} = [\mathcal{R}, T]$. For simplicity, we leave out the subscript frame ID l and points ID i. Let $p^L = (X, Y, Z)^T$ be the point in local frame

$$p^L = \mathcal{R}p + T \tag{A.1}$$

where $\mathcal{R} \in SO(3)$. the partial derivative of p^L over $\mathcal{R}$, T, p can be

$$\begin{aligned} \frac{\partial p^L}{\partial \mathcal{R}} &= -(\mathcal{R}p)^\vee \in \mathbb{R}^{3\times 3} \\ \frac{\partial p^L}{\partial \mathcal{T}} &= \mathcal{I} \in \mathbb{R}^{3\times 3} \\ \frac{\partial p^L}{\partial p} &= \mathcal{R} \in \mathbb{R}^{3\times 3} \end{aligned} \tag{A.2}$$

According to the projection function defined in Equation (22), the final partial derivatives can be easily derived by applying the chain rule.

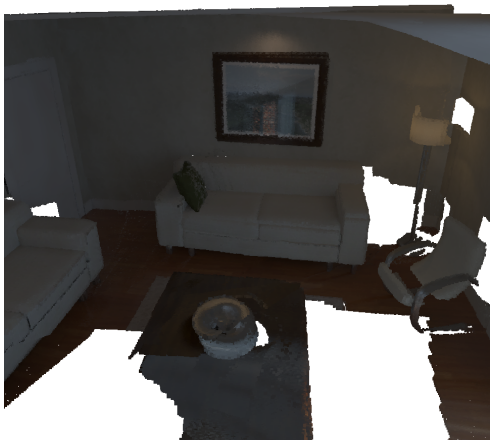
The variable nodes connected by a patch-to-patch edge e_τ^τ are the only camera poses, as the patch coefficients are regarded as constants in Equation (23). The following equations give the Jacobian matrix of e_τ^τ over the two camera poses. Only the first part of Equation (23) is provided as the second one can be derived similarly.

$$\begin{aligned} \frac{\partial e_\tau^\tau}{\partial \mathcal{R}^l} &= -((T^l)^T - (\tau_{a,i}^{l+1})^T \mathcal{R}^{l+1} - (\tau_{a,i}^{l+1})^T)(\mathcal{R}^l \tau_{n,i}^l)^\vee \\ \frac{\partial e_\tau^\tau}{\partial T^l} &= (\mathcal{R}^l \tau_{n,i}^l)^T \\ \frac{\partial e_\tau^\tau}{\partial \mathcal{R}^{l+1}} &= -(\tau_{a,i}^{l+1})^T (\mathcal{R}^{l+1})^T (\mathcal{R}^l \tau_{n,i}^l)^\vee \\ \frac{\partial e_\tau^\tau}{\partial T^{l+1}} &= -(\mathcal{R}\tau_{n,i}^l)^T \end{aligned} \tag{A.3}$$

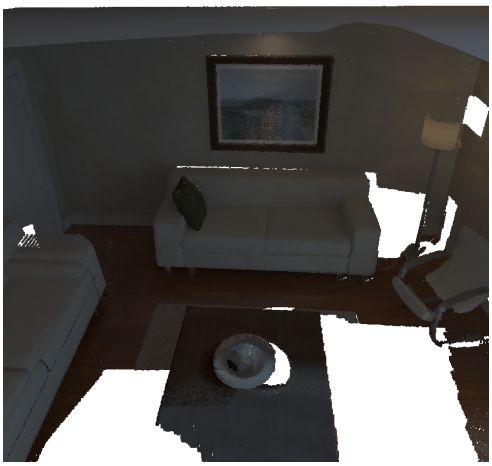
For the edge e_π^τ that connecting plane nodes and camera pose nodes via patches, the variables are the planes coefficients, and camera poses, but the measurements are from patches, as defined in Equation (24). The Jacobian matrix of this edge can be

(a) A model from Bundle Adjustment of [33]

(b) A model from our method



(c) Details of model by [33]



(d) Details of the our model

Figure 9: 3D reconstruction comparison on dataset ICL-NUIM: (a) The reconstructed model of batched BA from [33]. (b) The reconstructed model of proposed methods. (c) Some details from (a). (d) Some details from (b).

composed through the following partial derivatives.

$$\begin{aligned}
\frac{\partial e_\pi^\tau}{\partial \mathcal{R}^l} &= \tau^T \mathcal{R}^T (\pi^n)^\vee \\
\frac{\partial e_\pi^\tau}{\partial T^l} &= \pi_n \\
\frac{\partial e_\pi^\tau}{\partial \pi_n} &= (\mathcal{R}^l \tau_{a,i} + T^l)^T \\
\frac{\partial e_\pi^\tau}{\partial \pi_d} &= \mathcal{I};
\end{aligned} \tag{A.4}$$

[1] Barfoot, T. D., 2017. State Estimation for Robotics. Cambridge University Press.

[2] Bay, H., Tuytelaars, T., Van Gool, L., 2006. Surf: Speeded up robust features. Computer vision–ECCV 2006, 404–417.

[3] Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. Brief: Binary robust independent elementary features. Computer Vision–ECCV 2010, 778–792.

[4] Concha, A., Civera, J., 2017. RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system. CoRR abs/1703.00754.
URL http://arxiv.org/abs/1703.00754

[5] Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W., 2012. An evaluation of the rgb-d slam system. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, pp. 1691–1696.

[6] Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W., 2014. 3-d mapping with an rgb-d camera. IEEE Transactions on Robotics 30 (1), 177–187.

[7] Halber, M., Funkhouser, T. A., 2016. Structured global registration of RGB-D scans in indoor environments. CoRR abs/1607.08539.
URL http://arxiv.org/abs/1607.08539

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

[9] Hedau, V., Hoiem, D., Forsyth, D., 2009. Recovering the spatial layout of cluttered rooms. In: Computer Vision (ICCV), 2009 IEEE International Conference on. IEEE, pp. 1849–1856.

[10] Holz, D., Holzer, S., Rusu, R. B., Behnke, S., 2011. Real-time plane segmentation using rgb-d cameras. In: Robot Soccer World Cup. Springer, pp. 306–317.

[11] Horn, B. K. P., Apr 1987. Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. A 4 (4), 629–642.
URL http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629

(a) Results of Kintinuous

(b) Results of Elastic Fusion

(c) Results of our proposed method.

Figure 10: Comparison with real-time RGB-D SLAM Kintinuous and Elastic Fusion. On dataset SUN3D: (a) the result from Kintinuous; (b) the result of Elastic Fusion; (c) the output of our method.

[12] Huang, S., Wang, Z., Dissanayake, G., 2008. Sparse local submap joining filter for building large-scale maps. IEEE Transactions on Robotics 24 (5), 1121–1130.

[13] Kerl, C., Sturm, J., Cremers, D., 2013. Dense visual slam for rgb-d cameras. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, pp. 2100–2106.

[14] Khoshelham, K., Elberink, S. O., 2012. Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors 12 (2), 1437–1454.

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

[16] Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, pp. 1150–1157.

[17] Ma, L., Kerl, C., Stückler, J., Cremers, D., 2016. Cpa-slam: Consistent plane-model alignment for direct rgb-d slam. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, pp. 1285–1291.

[18] Martins, J. A. S., 2013. Mrslam-multi-robot simultaneous localization and mapping. Master's thesis.

[19] Mur-Artal, R., Tardós, J. D., 2017. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics 33 (5), 1255–1262.

[20] Ni, K., Steedly, D., Dellaert, F., 2007. Tectonic sam: Exact, out-of-core, submap-based slam. In: Robotics and Automation (ICRA), 2007 IEEE International Conference on. IEEE, pp. 1678–1685.

[21] Oehler, B., Stueckler, J., Welle, J., Schulz, D., Behnke, S., 2011. Efficient multi-resolution plane segmentation of 3d point clouds. Intelligent Robotics and Applications, 145–156.

[22] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., 2009. Ros: an open-source robot operating system. In: ICRA workshop on open source software. Vol. 3. Kobe, Japan, p. 5.

[23] Raposo, C., Antunes, M., Barreto, J. P., 2014. Piecewise-planar stereoscan: structure and motion from plane primitives. In: European Conference on Computer Vision (ECCV). Springer, pp. 48–63.

[24] Raposo, C., Lourenço, M., Antunes, M., Barreto, J. P., 2013. Plane-based odometry using an rgb-d camera. In: British Machine Vision Conference (BMVC).

[25] Rosten, E., Drummond, T., 2006. Machine learning for high-speed corner detection. Computer Vision–ECCV 2006, 430–443.

[26] Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2011. Orb: An efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE international conference on. IEEE, pp. 2564–2571.

[27] Salas-Moreno, R. F., Glocken, B., Kelly, P. H., Davison, A. J., 2014. Dense planar slam. In: Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on. IEEE, pp. 157–164.

[28] Stuckler, J., Behnke, S., May 2008. Orthogonal wall correction for visual motion estimation. In: Robotics and automation (ICRA), 2008 IEEE international conference on. pp. 1–6.

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

[30] Wang, J., Song, J., Zhao, L., Huang, S., 2018. A submap joining based rgb-d slam algorithm using planes as features. In: Field and Service Robotics. Springer, pp. 367–382.

[31] Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J., 2012. Kintinuous: Spatially extended kinectfusion. In: RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras.

[32] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., Davison, A. J., 2015. Elasticfusion: Dense slam without a pose graph. In: Robotics: science and systems. Vol. 11.

[33] Xiao, J., Owens, A., Torralba, A., 2013. Sun3d: A database of big spaces reconstructed using sfm and object labels. In: Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, pp. 1625–1632.

[34] Yang, M. Y., Förstner, W., 2010. Plane detection in point cloud data. In: Proceedings of the 2nd Int Conf on Machine Control Guidance, Bonn. Vol. 1. pp. 95–104.

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

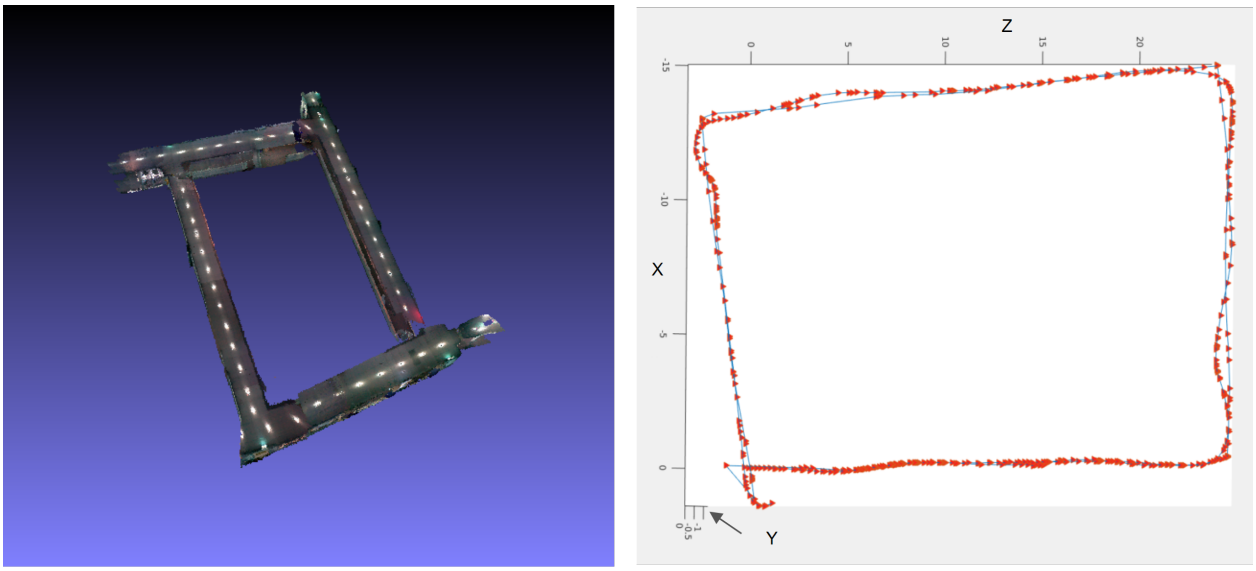


Figure 11: Our result using a dataset of a large corridor scenery collected by a robot. The corridor is well reconstructed as the planes on the floor and ceiling can be associated in our methods, and the plane constraints between different frames contribute to our camera pose estimation. (a) 3D model of the long corridor. (b) two-loop trajectory of the camera poses. The units are in meters.

[36] Zhao, L., Huang, S., Dissanayake, G., 2013. Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, pp. 24–30.

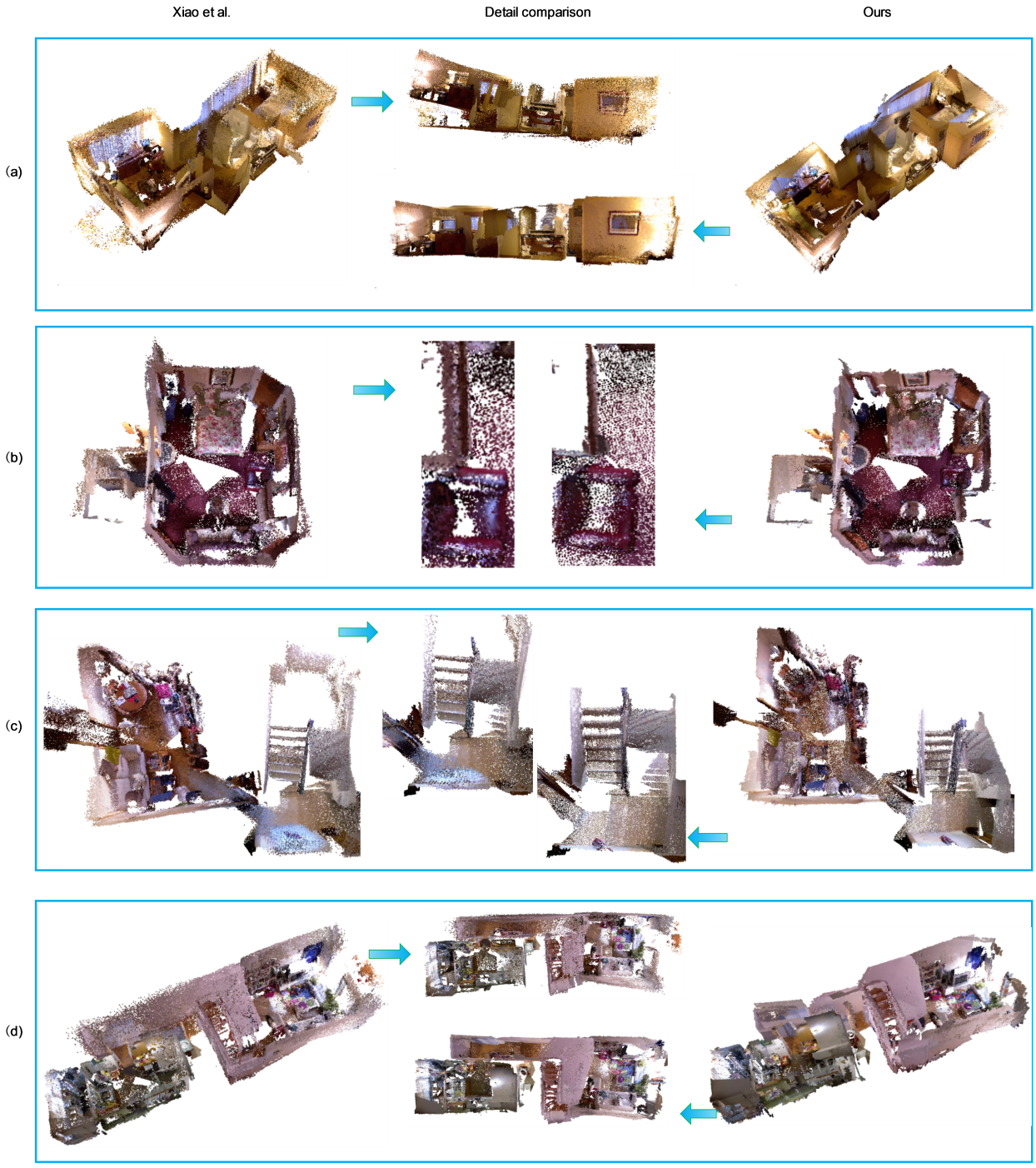


Figure 12: Comparison with traditional BA methods [33] on some publicly dataset from SUN3D. The first column shows the 3D reconstructed model from [33], the rightmost column present the 3D model from ours, the middle column compares the details (or different perspectives) from both models. (a) Hotel sf dataset. (b) Hotel std dataset. (c) Home md dataset. (d) Home ac dataset.