Plane-Based Optimization of Geometry and Texture for RGB-D Reconstruction of Indoor Scenes

Chao Wang and Xiaohu Guo
University of Texas at Dallas
800 W Campbell Rd, Richardson, TX 75080, USA
{chao.wang3, xguo}@utdallas.edu

Abstract

We present a novel approach to reconstruct RGB-D indoor scene with plane primitives. Our approach takes as input a RGB-D sequence and a dense coarse mesh reconstructed by some 3D reconstruction method on the sequence, and generate a lightweight, low-polygonal mesh with clear face textures and sharp features without losing geometry details from the original scene. To achieve this, we firstly partition the input mesh with plane primitives, simplify it into a lightweight mesh next, then optimize plane parameters, camera poses and texture colors to maximize the photometric consistency across frames, and finally optimize mesh geometry to maximize consistency between geometry and planes. Compared to existing planar reconstruction methods which only cover large planar regions in the scene, our method builds the entire scene by adaptive planes without losing geometry details and preserves sharp features in the final mesh. We demonstrate the effectiveness of our approach by applying it onto several RGB-D scans and comparing it to other state-of-the-art reconstruction methods.

1. Introduction

Online and offline RGB-D reconstruction techniques are developing fast in recent years with the prevalence of consumer depth cameras, and they have wide applications in many popular areas such as virtual reality (VR), augmented reality (AR), gaming and 3D printing. State-of-the-art online 3D reconstruction methods can capture indoor and outdoor scenes in the real-world environments efficiently with geometry details [19, 24, 25, 6, 21]. However, resulting 3D models of these methods are usually too dense with unsatisfying textures due to many reasons including noisy depth data, incorrect camera poses and oversmoothing in data fusion. These models can not be used directly in most applications without further refinement or post-processing.

In order to lower the density of reconstructed models and improve the structure quality, one typical strategy is to introduce plane primitives into front-end (such as tracking) or back-end (such as model refinement) of reconstruction pipeline, as typical indoor scenes are primarily composed of planar regions, especially buildings and houses with structure following Manhattan-world assumption. However, almost all existing planar reconstruction methods take into account only large planar regions such as walls, ceilings, floors and large table surfaces, and simply ignore and remove other objects with freeform surfaces no matter if they contain planar regions or not, such as various indoor furniture and objects on or adhering to large planes. These reconstructed models with only large planes are too simplified and lack fidelity that they are not applicable to many situations acquiring geometry details such as indoor roaming with decorations, gaming and various VR or AR applications. Moreover, one challenging problem in planar reconstruction of indoor scene is that geometry details are usually noisy because of noisy RGB-D raw data, and it is difficult and also time-consuming to extract plane primitives or other types of geometry priors from the scene while still preserving the original shape.

In this paper, we present a novel approach to reconstruct RGB-D indoor scene using planes and generate a lightweight and complete 3D textured model without losing geometry details. Our method takes as input a RGB-D sequence of indoor scene and a dense coarse mesh reconstructed by some online reconstruction method on this sequence. We firstly partition the entire dense mesh into different planar clusters, each of which represents a plane primitive. Next, we simplify the dense mesh based on the planar partitions into a lightweight mesh which preserves both large planar regions as well as small objects without losing geometry details. In order to generate texture mapping for mesh faces, we create texture patch for each plane and sample points on the plane, and run a global optimization process to maximize the photometric consistency of sampled points across frames by optimizing camera poses, plane parameters and texture colors. Finally, we optimize the mesh geometry by maximizing consistency between geometry and plane primitives, which further preserves sharp features of original scene such as edges and corners of plane intersections.

Our approach can be regarded as a back-end RGB-D reconstruction framework after online reconstruction process. Experiments show that our method exceeds state-of-the-art RGB-D planar reconstruction method in keeping geometry details and sharp features in the result lightweight 3D textured models.

2. Related work

Online and offline 3D reconstruction with consumer depth cameras have been widely studied in recent years. One major category of online large-scale 3D reconstruction methods utilizes volume data structure and volumetric fusion technique, like one of the earliest online method KinectFusion [18] and its subsequent variants such as scalable online reconstruction [4], VoxelHashing [19], Volume-shifting [23], BundleFusion [6] and InfiniTAM [21]. One important advantage of these volume-based methods is that it is fast and efficient to smooth noisy data scanned by depth cameras and easy to generate mesh with connected surface. However, a major limitation is that volumetric fusion oversmoothes the surface and removes the sharp features since it tends to average points and their colors in the same voxel position in the volume. Besides volume-based method, another strategy is point-based method, which utilizes surfels as basic data structure to represent the surface without connectivity [14, 24, 25]. However, it still suffers from oversmoothing problem since it also tends to make a moving average of points nearby in the space [24]. Moreover, error accumulates in surface representation and camera poses through the scanning process because of oversmoothing as well as other factors including noisy color and depth data.

In order to improve camera pose estimation and also the quality of surface, several works introduce planes into the reconstruction pipeline to estimate complicated surface with simple surface priors, especially in buildings or indoor scene with special structure features like Manhattan-world assumption. One category of methods introduce plane constraints into front-end of the reconstruction process to improve robustness of camera tracking. One of the earliest methods by Dou et al. [7] combines both plane and feature point correspondences to estimate camera poses in RGB-D scanning. A recent work by Hsiao et al. [12] uses very similar idea to introduce plane constraint into tracking in SLAM framework, and their result exceeds current state-of-the-art online 3D reconstruction methods in pose estimation. For offline reconstruction framework, a very recent work by Halber and Funkhouser [10] proposes a fine-to-coarse global registration algorithm on RGB-D data by combining planar relationship constraints with other types of constraints, and their method works efficiently on large-scale RGB-D data.

Another category of methods tries to utilize planes to better represent the surfaces of final model reconstructed from RGB-D data. Dzitsiuk et al. [8] introduce plane priors in real-time 3D reconstruction pipeline on mobile devices to lower the computation complexity, de-noise depth data and improve indoor structure. However, this method only refines several limited types of planes like walls, ceilings and floors and leaves other scanned data unchanged. Another impressive method 3DLite [13] is the latest method we found to reconstruct RGB-D model by planes and optimize face textures on them. This method generates lightweight model with optimized textures from an input dense and fused reconstruction. It proposes a frame-based plane detection technique to extract large planes from RGB-D mesh, and then globally optimizes camera poses and face textures on the planes. However, this method still cannot detect and simply removes all small planes and non-planar geometry details, whereas our method is to reconstruct the entire scene by planes without losing geometry details.

Planar reconstruction from point clouds is also a hot topic for decades, as point cloud data is prevalent and can be easily acquired via various tools such as Laser scanners or structure from motion (SfM) data. Planes are widely used in reconstruction of outdoor environments to fit building structure. Monszpart et al. [16] extract regular arrangement of planes from unstructured, large-scale and noisy building scans, and reconstruct the building model with complete and lightweight planes. However, it only detects large planar regions in the building framework and ignore details inside, so it is more like an architecture reconstruction method using planes. So is another method proposed by Mura et al. [17] that reconstructs only the permanent components (walls, ceilings and floors) of buildings by extracting planes on these components from all detected ones from input point cloud.

The two most common plane primitive detection strategies are RANSAC and region growing. RANSAC-based method is popular for its simplicity, scalability and probabilistic guarantees. However, it easily misses global scene-level structures and ignores the neighborhood regularity between points [16]. Therefore, RANSAC-based method are usually used on regular models such as CAD shapes and building structure [15]. Compared with RANSAC, region growing expands from seeds to neighbors and inherently detects parts that are connected, and is more suitable and widely used in plane detection of large-scale models. However, one important disadvantage is that region growing is not designed to detect planes on free-form shapes since it easily partitions curved surfaces into irregular parts and causes over-segmentation. Chauve et al. [3] propose one of the earliest methods to extract planes and build triangular mesh from noisy unstructured point cloud with planar regions. Boulch et al. [1] present a 3D reconstruction of a piecewise-planar surface from range images by regularizing the surface with respect to edges and corners. Oesau et al. [20] propose a planar shape detection and regularization method in tandem from raw point sets. Similarly, even though these methods do not ignore non-planar regions on purpose, they rely on region growing and still cannot preserve shape of curved surfaces.

3. Planar reconstruction pipeline

Our reconstruction pipeline takes a RGB-D sequence as input, and firstly uses some state-of-the-art online reconstruction such as BundleFusion [6] to reconstruct an initial dense mesh. Initially, we detect plane primitives by partitioning the mesh into different clusters, each of which represents a plane patch (Section 3.1). The following step is to simplify the dense mesh based on the planar partition (Section 3.2). Next, we sample points on each plane in the mesh and create texture patch for the plane patches, and optimize camera poses, plane parameters and texture colors to maximize the color consistency across frames (Section 3.3). Finally, we optimize the mesh geometry to maximize consistency between vertices on the mesh and corresponding plane primitives (Section 3.4).

3.1. Mesh planar partition

Unlike existing planar reconstruction methods which only take into account large planar regions, we aim to partition the entire mesh into plane primitives to include all geometry details. As we introduce in Section 2, two most common plane detection method is RANSAC and region growing. RANSAC-based detection is efficient but easily misses global scene-level structures, while region growing is more robust in detecting planes under regularity but still is not designed to detect planes in freeform shapes with curved surfaces. 3DLite [13] proposes a frame-based plane detection method to extract large planes from RGB-D mesh. However, this method still cannot detect small planar regions or non-planar surfaces and it simply removes them in the final mesh. Moreover, the frame-by-frame plane detection and merging scheme used in 3DLite is time-consuming and easy to accumulate errors so it has to utilize many further optimization steps for robustness.

In our approach we refer to a state-of-the-art surface partition algorithm proposed by Cai et al. [2]. This method proposes a new principle component analysis (PCA) based energy, whose minimization leads to an optimal piecewise-linear planar approximation of the entire surface with high quality. After an input mesh is partitioned into clusters, each cluster is attached with a plane proxy ϕ_i defined by the centroid $\mathbf{c}_i$ and normal $\mathbf{n}_i$ as the smallest eigenvector direction from the covariance matrix of the cluster. In order to detect planes more sensitively and regularize the compactness of planar clusters, in our experiments we lower the coefficient α used in Eq. (4) of the paper [2] from 10^{-15} to 10^{-20}. Figure 1 (left) shows planar partition result on a dense mesh reconstructed by online reconstruction system VoxelHashing [19] with groundtruth camera poses.

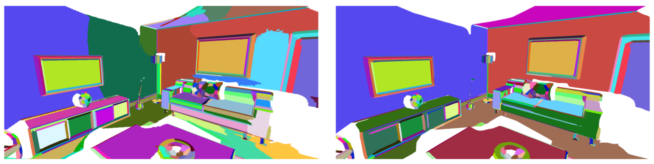
Figure 1. Plane partition result on a mesh without plane merging (left) and with merging (right) on the 'lr kt2n' model from ICL-NUIM dataset. Notes that small neighbor plane patches on the walls, floors and ceilings are merged into large planes.

After we get the initial planar partition, we run a further plane merging step to merge **adjacent** planes together into large ones (Figure 1 right). The reason is that, the reconstructed mesh on noisy RGB-D data always contains noise such as bumpy points on planar regions. Because of this, a large planar region is possibly partitioned into several different small clusters instead of only one large cluster (see the walls in Figure 1 left). Our plane merging criteria is like this: two adjacent planes i and j can be merged together if they satisfy all following conditions:

(1) Angle between two plane normals are small: $\theta(\mathbf{n}_i, \mathbf{n}_j) < \epsilon_{\mathbf{n}}$;

(2) Average distance between two planes is small: $Avgdis(i,j) < \epsilon_d,\ \ Avgdis(j,i) < \epsilon_d$;

(3) Angle between the ray connecting two plane centers $\mathbf{r}_{ij} = \mathbf{c}_i - \mathbf{c}_j$ and each plane normal is as close as possible to $90°$: $|\cos(\theta(\mathbf{r}_{ij}, \mathbf{n}_i))| < \epsilon_{\mathbf{c}},\ \ |\cos(\theta(\mathbf{r}_{ij}, \mathbf{n}_j))| < \epsilon_{\mathbf{c}}$.

Here $\theta(\cdot,\cdot)$ is the angle between two vectors, and $Avgdis(i,j)$ is the average distance between all vertices in cluster i and plane j. In our experiment we use $\epsilon_{\mathbf{n}} = 8°, \epsilon_d = 0.05, \epsilon_{\mathbf{c}} = \cos(80°)$. The first two conditions are from the plane merging method by Dzitsiuk et al. [8]. We add the third rule to get rid of special cases that two adjacent planes are almost 'overlapped' in the noisy mesh, and only merge neighbor planes that are on one side of each other.

3.2. Mesh simplification

After partitioning the mesh into plane clusters, we simplify the mesh based on clusters to create a lightweight mesh for further optimization. Note that even though we already have a model composed of planes, we still choose to create a mesh by simplifying the original dense mesh instead of using some mesh generation algorithm (such as Delaunay triangulation) on planes which strategy appears in most existing methods [3, 13, 15]. The reason is that, we found that it is difficult and also time-consuming to create correct connectivity from complicated plane interceptions in a noisy model, especially an indoor reconstruction mesh containing various geometry objects with free-form shapes. Therefore, our strategy is to efficiently create an initial lightweight mesh through simplifying the original dense mesh based on planes, and further optimize its geometry to fit the planes (Section 3.4).

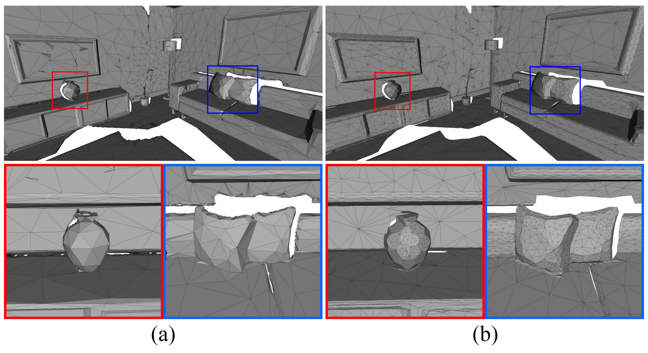

Figure 2. Simplified mesh by standard QEM (a) and our method (b) on the scan 'lr kt2n' from ICL-NUIM dataset. Note for sharp features such as plane boundaries, and curved surfaces such as vase and pillows.

During mesh simplification, we choose the classic quadric error metric (QEM) based surface simplification [9]. The core of QEM is to contract edges by minimizing a point-to-plane energy, which suits our purpose to preserve plane shapes very well. Moreover, QEM is simple and efficient and also tends to preserve sharp features of original model, which is also what we need in the simplified mesh. The difference is that, unlike a standard QEM process which puts all edges into a minimum heap based on contraction energy and contracts them globally, our simplification process works cluster-by-cluster and is composed of two steps: (1) fix all cluster boundary edges and simplify only inner-cluster edges of each cluster separately; (2) fix simplified inner-cluster edges and simplify all boundary edges of clusters. Through this two-step process, we can reduce unimportant information inside planes while still preserving important sharp features which usually appear along plane boundaries. Another consideration is that cluster sizes are different from each other, but we prefer every cluster contains almost the same number of edges after simplification. That is, large planes contain large triangles and vise versa. Then, simplifying cluster-by-cluster with a same contraction target is better to make small clusters denser to preserve their shapes, as they are usually the partition of curved surfaces. Figure 2 shows comparison between standard QEM (keep mesh boundary) and our method on the same model used in Figure 1. The two simplified models have exactly the same number of faces 41K while mesh QEM model has 30K points, larger than our model with 22K points. Obviously QEM oversimplifies plane boundaries and curved surfaces while our method keeps the sharp features and geometry details better.

3.3. Plane, camera pose and texture optimization

After we get the simplified mesh with plane partition, we run an optimization process to maximize the photometric consistency for the mesh geometry between frames. Before optimization, we still need to do some pre-processing on the mesh.

The first work is to generate an initial texture mapping for all the faces of the mesh. In our method we create a 2D texture patch for each 3D plane on the mesh. Even though there are many mature parameterization methods to generate 2D texture patches for 3D mesh surface, we use a very simple and efficient one. Considering that vertices in each cluster on the mesh are already near co-planar, we simply project them onto the plane to get the 2D patch, and then sample grid points inside the patch boundary to get texel points. In our experiment we use a fixed sampling density 0.0025m. That is, 1 meter in global space corresponds to $1.0/0.0025 = 400$ pixels in the texture image. Clearly, each texel is located inside some projected triangluar face from the mesh. Then, we compute each texel's barycentric coordinates inside its corresponding 2D face, and use them to compute the texel's corresponding 3D point $\mathbf{p}$ in global space by simply applying the same barycentric coordinates onto the three vertices of the face in 3D space. These 3D texel points will be used as the major elements during optimization process.

Another thing to mention is about the keyframes selected from RGB-D frames. To reduce time complexity and increase texture quality, we follow the similar idea of color map optimization by Zhou and Koltun [26] to select only sharp frames in every interval. Similar to their method, we quatify the blurriness of each image with the metric by Crete et al. [5]. The difference is that, instead of selecting keyframe for every 1 to 5 seconds in [26], we simply select the sharpest frame in every 5 or 10 frames depending on the data.

The input in our optimization process is color images $\{\mathbf{I}_i\}$ and depth images of keyframes, all texels' 3D points $\{\mathbf{p}\}$ sampled on the mesh, initial camera poses $\mathbf{T} = \{\mathbf{T}_i\}$ (global to camera space) and initial plane parameters $\Phi = \{\phi_j\}$. During the optimization, we maximize the photo consistency of 3D texels' projections on corresponding planes across frames by optimizing camera poses, plane parameters and texture colors. More specifically, our objective function is

$$E_{tex}(\mathbf{T},\Phi,\mathbf{C},\mathbf{F}) = E_c(\mathbf{T},\Phi,\mathbf{C},\mathbf{F})+\lambda_1 E_p(\Phi)+\lambda_2 E_s(\mathbf{F}), \tag{1}$$

where E_c is photometric consistency energy, E_p is constraint for planes, E_s is constraint for non-rigid correction offsets for color images introduced in [26], and λ_1 and λ_2 are coefficients to balance terms. In our experiment we use λ_1 such that initial values of $E_c = \lambda_1 E_p$ and $\lambda_2 = 0.1$.

Photometric consistency term. The photometric energy is designed to measure the photometric error between color of each texel's projection point on its corresponding plane and its target color across frames:

$$E_c(\mathbf{T},\Phi,\mathbf{C},\mathbf{F}) = \sum_i \sum_{\mathbf{p}\in\mathbf{P}_i} ||C(\mathbf{p}) - \mathbf{I}_i(\mathbf{F}_i(\pi(\mathbf{T}_i\mathbf{q})))||^2, \tag{2}$$

where $C(\mathbf{p})$ is the target color for $\mathbf{p}$, and $\mathbf{P}_i$ is set of all visible 3D texels in frame i, and π is the perspective projection on homogeneous coordinate $\mathbf{v} = (v_0, v_1, v_2, v_3)^\top$:

$$\pi(\mathbf{v}) = (\frac{v_0 * f_x}{v_2} + c_x, \frac{v_1 * f_y}{v_2} + c_y)^\top \tag{3}$$

where c_x, c_y, f_x, f_y are principal point and focal lengths from camera intrinsic matrix, respectively. $\mathbf{F} = \{\mathbf{f}_{i,l}\}$ in Eq. (2) is the set of non-rigid correction fuctions of control vertices over color image $\mathbf{I}_i$ [26]:

$$\mathbf{F}_i(\mathbf{u}) = \mathbf{u} + \sum_l \delta_l(\mathbf{u})\mathbf{f}_{i,l}, \tag{4}$$

where δ_l is the basis functions for bilinear interpolation, $\mathbf{f}_{i,l}$ is 2D correction vector for lth control vertex in the orthogonal grid in color image of frame i. In our experiment we follow the same parameters from [26] to use 20×16 grid on each image. $\mathbf{q}$ in Eq. (2) is the projection point from $\mathbf{p}$ onto its corresponding plane $\phi(\mathbf{p})$ represented by 3D normal $\mathbf{n_p}$ and a scalar $w_\mathbf{p}$:

$$\mathbf{q} = \mathbf{p} - (\mathbf{p}^\top \mathbf{n_p} + w_\mathbf{p})\mathbf{n_p}, \tag{5}$$

Plane constraint term. Plane constraint term is to minimize the sum of distances from 3D texel points to their corresponding planes:

$$E_p(\Phi) = \sum_\mathbf{p} ||\mathbf{p}^\top \mathbf{n_p} + w_\mathbf{p}||^2 \tag{6}$$

Offset constraint term. The offset constraint is to minimize the magnitude of offset vectors:

$$E_s(\mathbf{F}) = \sum_i \sum_l \mathbf{f}_{i,l}^T \mathbf{f}_{i,l} \tag{7}$$

In order to minimize the objective function in Eq. (1), we follow the similar alternating optimization strategy in [26]. The basic idea is to alternate between optimizing different variables with some others fixed. In each iteration we firstly optimize $\mathbf{C}$ and fix all the others, and next optimize Φ and fix the others, and finally optimize $\mathbf{T}$ and $\mathbf{F}$ and fix the others. When optimizing $\mathbf{C}$, the problem becomes a linear system with closed form solution that $C(\mathbf{p})$ is the average color of all projected pixels associated with $\mathbf{p}$ in all visible frames. When optimizing Φ, we use standard Gauss-Newton method to update the 4 parameters of each plane directly. Note that the optimization of each plane is independent with others so we can solve them in parallel. Optimizing $\mathbf{F}$ is similar as optimizing Φ that we update all relevant variables directly. When optimizing $\mathbf{T}$, we parameter each pose $\mathbf{T}_i$ by 6 parameters (3 for rotation, 3 for translation) as the incremental transformation, and locally linearize each pose around its last updated value. Similarly, optimization of each $\mathbf{T}_i$ and $\mathbf{F}_i$ are also independent with other frames and can be solved in parallel. Figure 3 shows textures on a mesh before and after our optimization process. Before optimization, we use average color from all visible frames for each texel. Our optimization process can reduce noise and make texture clearer.

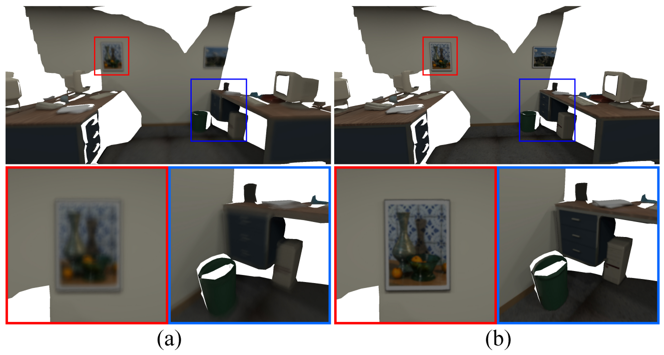

Figure 3. Mesh with textures before and after texture optimization on the simplified model of sequence 'of kt2' from ICL-NUIM dataset [11]. (a) Texture before optimization. Each texel's color is the average color from all visible color frames. (b) Texture after optimization.

3.4. Geometry optimization

The final step is to optimize the mesh geometry to fit the planes as close as possible. The fused mesh reconstructed from RGB-D data always contain noise or oversmoothed surfaces, such as bumpy surfaces on planar regions and smoothed borders which suppose to be sharp features. By optimizing mesh geometry to fit the optimized planes, we can reduce noise from mesh surface and sharpen geometry features.

In order to optimize the consistency between geometry and planes, our method is to maximize the consistency between mesh vertices in each cluster and their corresponding planes. As we introduced in Section 3.3, each 2D texel is located inside a triangular face's projection. During the optimization, we utilize the initial barycentric relationship between each texel and its corresponding face, and try to preserve this relationship between texel points' projections on planes and the optimized vertices in each face. Specifically, we want to minimize the following energy function w.r.t. all vertices $\mathbf{V} = \{\mathbf{v}_i\}$:

$$E_{vert}(\mathbf{V}) = E_g(\mathbf{V}) + \lambda_3 E_t(\mathbf{V}). \tag{8}$$

Here E_g is the geometry consistency term

$$E_g(\mathbf{V}) = \sum_{\mathbf{p}} ||\mathbf{q} - \sum_{i=0}^{2} b_{\mathbf{p},i}\mathbf{v}_{f_{\mathbf{p}},i}||^2, \tag{9}$$

where $\mathbf{q}$ is the projection from 3D texel point $\mathbf{p}$ onto its corresponding plane as described in Eq. (5), $f_{\mathbf{p}}$ is index of the face $\mathbf{p}$ corresponds to, $\mathbf{v}_{f_{\mathbf{p}},i}$ is the ith vertex of face $f_{\mathbf{p}}$, and $b_{\mathbf{p},i}$ is $\mathbf{p}$'s barycentric coordinate corresponding to the ith vertex in face $f_{\mathbf{p}}$.

Another term E_t in Eq. (8) is a regularization term to minimize the difference between each vertex and the mass center of all its neighbors:

$$E_t(\mathbf{V}) = ||\mathbf{L}\mathbf{X}||_F^2. \tag{10}$$

Here $\mathbf{X} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]^\top$ is matrix of target vertices we want to compute, with n the number of vertices on the mesh. $\mathbf{L}$ is $n \times n$ matrix denoting the discrete graph Laplacian matrix based on the connectivity of the mesh, and its elements are

$$\mathbf{L}_{ii} = 1, \quad \mathbf{L}_{ij} = \begin{cases} -\frac{1}{|N(i)|} & j \in N(i) \\ 0 & j \notin N(i) \end{cases}, \tag{11}$$

where $N(i)$ is the set of $\mathbf{v}_i$'s neighbor vertices on the mesh. That is, we want to minimize the difference between each optimized vertex and the average of its neighbor vertices. The term E_t is added to ensure that problem in Eq. (8) has valid solutions. λ_3 in Eq. (8) is for balancing the two terms and we simply use $\lambda_3 = 1.0$.

The problem in Eq. (8) is actually a sparse linear system and can be solved by Cholesky decomposition efficiently. Figure 4 shows result mesh with optimized geometry on a scan from BundleFusion dataset [6]. Even though the input dense model is oversmoothed on sharp feature places, our method can preserve the sharp features in the final lightweight mesh.

4. Results

We tested our method on 10 scans from three popular RGB-D datasets: 6 models from BundleFusion [6] (the first 6 rows in Table 1), 2 from ICL-NUIM [11] (the following 2 rows in Table 1) and 2 from TUM RGB-D dataset [22] (the last 2 rows in Table 1). According to Huang et al. [13], the online BundleFusion code possibly generates some artifacts in the resulting reconstruction. So in our experiment, we follow the same idea of [13] to run VoxelHashing code [19] on each RGB-D sequence to reconstruct our input dense mesh using groundtruth poses (BundleFusion dataset provides poses computed by BundleFusion system). Table 1 shows quantitative results of each scan and our result models. Note that the number of faces or vertices of each result model is only 1%-3% of that of original dense model. Figure 5 and 6 show more qualitative results of two scans. These figures show that our method can generate a lightweight mesh with preserved sharp features and good face textures.

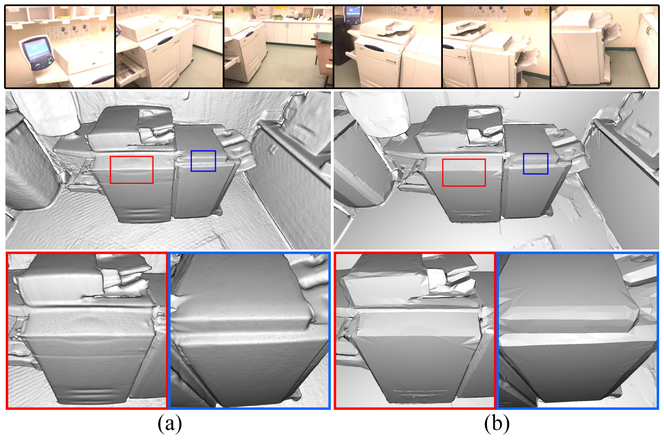

Figure 4. Comparison between reconstructed mesh with or without geometry optimization on model 'copyroom' from BundleFusion dataset [6]. First row shows selected input color images of the scene. Note for the sharp edges between planes on the printer. (a) Fused dense mesh from BundleFusion system. Note that sharp features are oversmoothed. (b) Our simplified mesh after geometry optimization. Note that sharp features are clear.

For time analysis, we implemented our method in C++ and tested on a desktop with Intel Core i7 2.5GHz CPU and 16 GB memory. The running time on each scan is shown in Table 1. The time data in the table is total time of our complete pipeline, of which plane partition (Section 3.1) and mesh simplification (Section 3.2) steps both take about 2 min on each model, while geometry optimization (Section 3.4) takes less than 10 seconds, and the majority rest is for plane, camera pose and texture optimization (Section 3.3). Note that our code is CPU version only and it currently does not contain any parallel acceleration technique in all steps. As we describe in Section 3.3, optimization of each plane is independent with each other and the process can be run in parallel. So is the same for the optimization of each camera pose and correction function in each frame. Moreover, we found that the majority of time in the optimization process is spent on computing the Jacobian matrix in Gaussian Newton algorithm, which can be greatly accelerated using GPU since each texel's computation process is also independent with all the others.

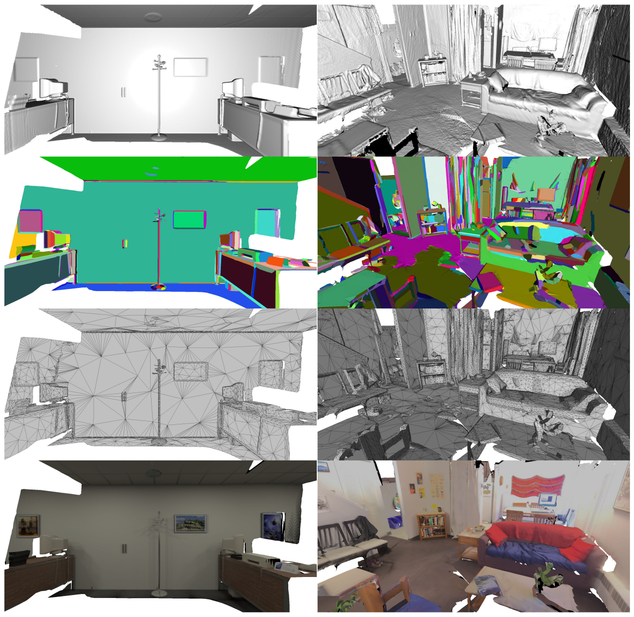

Figure 5. Our result on two RGB-D scans: 'of kt2' (left column) from ICL-NUIM dataset [11] and 'apt0' (right column) from BundleFusion dataset [6]. First row: input dense model. Second row: plane partition. Third row: result mesh. Forth row: mesh with textures.

For comparison, we compare our method with two state-of-the-art systems: BundleFusion [6] and 3DLite [13]. BundleFusion generates a dense and fused reconstruction in real-time, and 3DLite is similar to our method that it takes as input a dense reconstruction from BundleFusion, generates lightweight mesh with face textures by extracting large planar regions as geometry and optimizing texture on it. Figure 6 shows comparison of mesh with face textures between these methods. BundleFusion models are dense with oversmoothed sharp features and coarse textures. 3DLite models are neat and clear with sharp textures. However, it extracts only large planar regions from the scene as the final mesh, and obviously misses almost all geometry objects on or adhering to planes including both large structured ones (such as book shelf in the 1st row of Figure 6) and freeform objects with curved surfaces (such as objects on the table in the 2nd row of Figure 6). Our method exceeds them in solving their aforementioned problems. Moreover, 3DLite also generates many misaligned artifacts in the face textures while our method is better in these relevant places (such as the map on the wall in the 3rd row of Figure 6).

Table 1. Quantitative data of RGB-D scans and our results. Here $|V|$ is number of vertices, $|F|$ is the number of faces, $|K|$ is the number of keyframes, t is the total running time of our entire pipeline in seconds. The bottom row 'fr3/loh' is shorted for 'fr3/long_office_household' model.

Scan	Input			Result		
	$\|V\|$	$\|F\|$	$\|K\|$	$\|V\|$	$\|F\|$	t(s)
copyroom	3.70M	7.28M	895	55.2K	104K	1850
apt0	7.83M	15.4M	860	84.6K	160K	2125
office0	5.71M	11.3M	616	68.5K	130K	1439
office1	6.03M	11.9M	573	69.1K	129K	1331
office2	5.63M	11.0M	700	73.6K	135K	1886
office3	6.36M	12.6M	763	56.7K	108K	1972
of kt2	1.20M	2.36M	176	14.9K	27.4K	986
lr kt2n	1.14M	2.25M	176	22.1K	41.9K	1128
fr2/desk	1.37M	2.69M	372	37.6K	73.4K	787
fr3/loh	2.42M	4.75M	243	43.0K	83.7K	576

Limitations. While our method can generate lightweight textured mesh with sharp features and geometry details preserved, it still contains several limitations as we found in experiments. Firstly, our face textures are not as sharp as 3DLite's since the latter introduces many techniques to optimize texture, such as texture sharpening and color correction across frames. However, these steps are very time-consuming and possibly takes hours, according to the running time data described in the 3DLite paper [13]. We plan to find a similar but faster way to further optimize textures. Moreover, our method cannot fill holes and gaps that always appears in the RGB-D scans, while 3DLite can generate a complete geometry from extracted large planes by extrapolating existing planes and filling holes. However, it is still a challenging problem to complete the geometry of a noisy reconstruction without removing any geometry details. Additionally, our texture optimization process is similar to Zhou and Koltun [26] which relies on dense photometric error, and sensitive to initial input during camera pose optimization and easy to end in local minima [13]. Therefore, if large error exists in initial camera poses generated from BundleFusion system, our results may contain misaligned face textures.

5. Conclusion

We have presented a novel approach to generate lightweight reconstruction with clear face texture from an initial 3D reconstruction with dense and fused mesh. Unlike existing methods which only detect large planar regions in the scene, our method detects planes from all objects and partitions the entire mesh by planes. Then, we simply the dense mesh based on planar partition, and optimize planes, camera poses and face textures to maximize the photometric

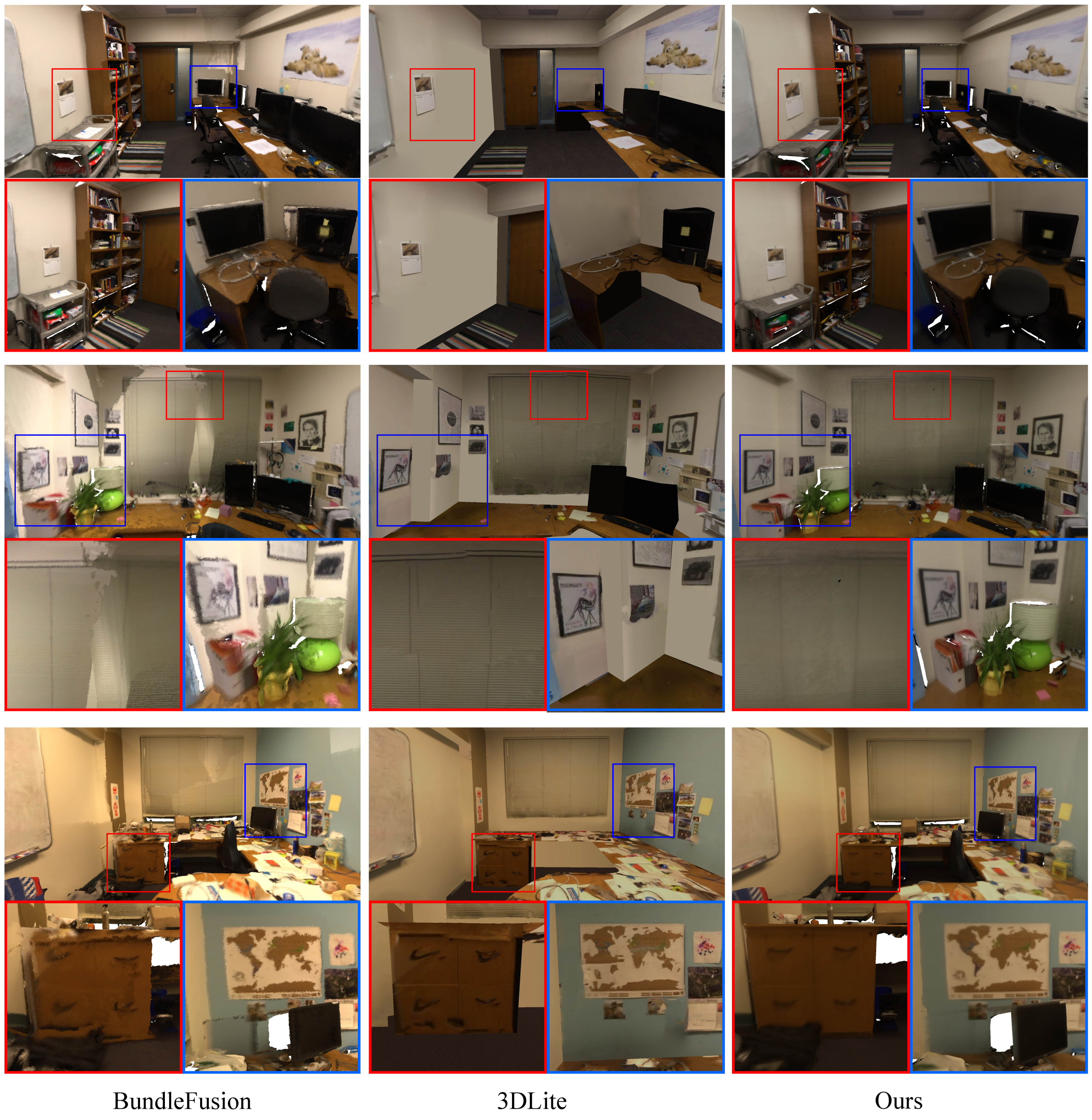

Figure 6. Result comparison between BundleFusion [6], 3DLite [13] and ours on two scans 'office0' (first two rows) and 'office3' (third row) both from BundleFusion dataset. The color texture in BundleFusion models is default vertex colors fused from the reconstruction system, while others are optimized face texture.

consistency across frames, and finally optimize mesh geometry to maximize plane-geometry consistency. Experimental results demonstrate that our reconstruction can preserve sharp features and geometry details in the mesh very well. We believe that our method can be applied in relevant situations acquiring textured indoor scene reconstruction without missing geometry details. In the future, we plan to improve our method by solving limitations aforementioned, and increase the efficiency of our method by introducing GPU acceleration or other related techniques.

References

[1] A. Boulch, M. de La Gorce, and R. Marlet. Piecewise-planar 3d reconstruction with edge and corner regularization. In *Computer Graphics Forum*, volume 33, pages 55–64. Wiley Online Library, 2014.

[2] Y. Cai, X. Guo, Y. Liu, W. Wang, W. Mao, and Z. Zhong. Surface approximation via asymptotic optimal geometric partition. *IEEE transactions on visualization and computer graphics*, 23(12):2613–2626, 2017.

[3] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1261–1268. IEEE, 2010.

[4] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(4):113, 2013.

[5] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Human vision and electronic imaging XII*, volume 6492, page 64920I. International Society for Optics and Photonics, 2007.

[6] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

[7] M. Dou, L. Guan, J.-M. Frahm, and H. Fuchs. Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera. In *Asian Conference on Computer Vision*, pages 94–108. Springer, 2012.

[8] M. Dzitsiuk, J. Sturm, R. Maier, L. Ma, and D. Cremers. De-noising, stabilizing and completing 3d reconstructions on-the-go using plane priors. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3976–3983. IEEE, 2017.

[9] M. Garland. Quadric-based polygonal surface simplification. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1999.

[10] M. Halber and T. Funkhouser. Fine-to-coarse global registration of rgb-d scans. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[11] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[12] M. Hsiao, E. Westman, G. Zhang, and M. Kaess. Keyframe-based dense planar slam. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5110–5117. IEEE, 2017.

[13] J. Huang, A. Dai, L. Guibas, and M. Nießner. 3dlite: towards commodity 3d scanning for content creation. *ACM Transactions on Graphics*, 2017, 2017.

[14] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 1–8. IEEE, 2013.

[15] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics (TOG)*, volume 30, page 52. ACM, 2011.

[16] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103–1, 2015.

[17] C. Mura, O. Mattausch, and R. Pajarola. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In *Computer Graphics Forum*, volume 35, pages 179–188. Wiley Online Library, 2016.

[18] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[19] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):169, 2013.

[20] S. Oesau, F. Lafarge, and P. Alliez. Planar shape detection and regularization in tandem. In *Computer Graphics Forum*, volume 35, pages 203–215. Wiley Online Library, 2016.

[21] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray. Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*, 2017.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[23] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[24] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[25] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[26] Q.-Y. Zhou and V. Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014.