Doctoral Dissertation

Color and 3D Semantic Reconstruction of Indoor Scenes from RGB-D Streams

Junho Jeon (전 준 호)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2019


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

RGB-D 영상을 통한 실내 환경의 색상 및 3D 의미론적 복원

Color and 3D Semantic Reconstruction of Indoor Scenes from RGB-D Streams

Color and 3D Semantic Reconstruction of Indoor Scenes from RGB-D Streams

by

Junho Jeon
Department of Computer Science and Engineering
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of philosophy in the Computer Science and Engineering

Pohang, Korea
11. 20. 2018
Approved by
Seungyong Lee
Academic advisor

Color and 3D Semantic Reconstruction of Indoor Scenes from RGB-D Streams

Junho Jeon

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH

11. 20. 2018

Committee Chair 이 승 용 (Seal)

Member 홍 기 상 (Seal)

Member 윤 국 진 (Seal)

Member 조 민 수 (Seal)

Member 곽 수 하 (Seal)

DCSE 전 준 호. Junho Jeon
20120842
Color and 3D Semantic Reconstruction of Indoor Scenes from RGB-D Streams,
RGB-D 영상을 통한 실내 환경의 색상 및 3D 의미론적 복원
Department of Computer Science and Engineering , 2019, 104p, Advisor : Seungyong Lee. Text in English.

ABSTRACT

Despite recent successes of 3D reconstruction, the majority of researches mainly focus on acquiring the precise geometric representation. Even though many computer graphics applications need more than just scene geometry such as lighting condition, surface materials and semantic labels of the scene, existing 3D reconstruction algorithms leave such auxiliary information behind their consideration. Nevertheless, investigating the reconstruction of such auxiliary information can provide richer user experience to the virtual/augmented reality applications. In this thesis, we present three auxiliary reconstruction techniques: intrinsic image decomposition, texture map generation for 3D reconstruction, and semantic reconstruction.

First, we present a novel image model for handling textures in intrinsic image decomposition, which enables us to produce high-quality results even with simple constraints. We also propose a novel constraint based on surface normals obtained from an RGB-D image to promote local and global consistency on the shading image. As a result, the proposed method can produce superior decomposition

results to existing approaches.

We also present a novel texture map generation for 3D reconstructed scenes. To represent the color information of 3D reconstruction, the proposed method uses texture mapping with simplified mesh instead of millions of vertex colors. We acquire an accurate texture map by optimizing the texture coordinates of the 3D model to maximize the photometric consistency among multiple keyframes. Our method can generate a texture map in a few tens of seconds for a large 3D model, such as a whole room.

Lastly, we present a 3D semantic reconstruction technique for generating semantically segmented triangular meshes of reconstructed 3D indoor scenes. During dense surface reconstruction, our framework predicts 2D CNN-based semantic segmentation of multiple frames and integrates them into an uniform voxel grid via volumetric semantic fusion. The proposed framework equipped with a structure-aware adaptive integration and CRF-based regularization can easily generate a high-quality 3D scene model with precise and dense (i.e., per-vertex) semantic labels.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

I. Introduction

1.1 Motivation

In recent years, as the computer graphics applications including virtual and augmented reality have got attention from consumers, a demand for complex and realistic 3D contents has exploded drastically. However, since manually generating the realistic 3D scenes is time and cost-consuming, direct generation of 3D models from a real-world environment, namely 3D reconstruction, has been researched by many research groups.

From a real-world object or scene, 3D Reconstruction can generate the virtual 3D models which can be rendered by computer graphics applications. Based on the early computer vision algorithms such as stereo vision and structure from motion, 3D reconstruction methods using multiple color images has been researched [1, 2, 3]. More recently, thanks to the consumer depth sensors like Microsoft Kinect and Structure sensor, sophisticated geometric reconstruction techniques [4, 5, 6] have been rapidly developed. Nowadays, densely reconstructed 3D geometry in the form of volume or triangle mesh can be acquired in real-time using consumer depth sensors, even for the large-scale scenes such as a floor or a whole building.

Despite those successes of 3D reconstruction, the majority of researches mainly focus on acquiring the precise geometric representation. Even though many computer graphics applications require more than a just scene geometry such as lighting condition, surface materials and semantic labels of the scene, existing 3D reconstruction algorithms [4, 5, 6] leave such auxiliary information out of their consideration.

(a) intrinsic images (b) color reconstruction (c) semantic reconstruction

Figure 1.1: Our 3D color and semantic reconstruction of indoor scenes.

In this thesis, we present three reconstruction techniques for auxiliary information: intrinsic image decomposition, texture map generation for 3D reconstruction, and semantic reconstruction. Firstly, because the reconstructed 3D models are rendered in different lighting conditions according to their application, there is a strong demand for intrinsic image decomposition which decomposes a given color image into the reflectance and shading images. While intrinsic image decomposition has been studied extensively during the past a few decades, it is still a challenging problem. This is partly because commonly used constraints on shading and reflectance are often too restrictive to capture an important property of natural images, i.e., rich textures.

Secondly, for an efficient rendering of large-scale 3D reconstructed models, texture mapping of the model is highly required. However, most 3D reconstruction methods [4, 5, 6] recover the color information in the form of vertex colors, and a tremendous number of reconstructed vertices requires severe rendering overhead. Moreover, the recovered vertex colors suffer from ghosting artifacts as they are acquired by simple blending of imprecisely aligned color images.

Lastly, existing 3D reconstruction method builds a single connected triangle mesh for the entire scene rather than the disjoint object meshes. It may critically reduce the user experience of virtual reality or augmented reality applications

which demand active interaction with the objects in the environment. Therefore, 3D segmentation method that precisely separates the reconstructed 3D meshes into the disjoint object meshes in accordance with their semantic object classes is a useful technique for the 3D scene reconstruction.

For more general and accurate intrinsic image decomposition, we present a novel image model for handling textures in decomposition, which enables us to produce high-quality results even with simple constraints.

We also propose a novel constraint based on surface normals obtained from an RGB-D image. Assuming Lambertian surfaces, we formulate the constraint based on a locally linear embedding framework to promote local and global consistency on the shading layer. We demonstrate that combining the novel texture-aware image model and the novel surface normal based constraint can produce superior results to existing approaches

For the efficient and clear texture map generation, we present a novel method for generating texture maps for 3D geometric models reconstructed using consumer RGB-D sensors. Our method generates a texture map for a simplified 3D mesh of the reconstructed scene using spatially and temporally sub-sampled key frames of the input RGB stream. We acquire an accurate texture map by optimizing the texture coordinates of the 3D model to maximize the photometric consistency among multiple keyframes. We show that the optimization can be performed efficiently using GPU by exploiting the locality of texture coordinate manipulation. Experimental results demonstrate that our method can generate a texture map in a few tens of seconds for a large 3D model, such as a whole room.

For the semantic segmentation of the 3D reconstructed scenes, we propose the semantic reconstruction framework that generates a dense semantic segmentation of 3D indoor scene meshes via a volumetric semantic fusion of multiple

2D semantic predictions in the reconstruction process. Our method integrates the results of CNN-based 2D semantic segmentation method that is applied to the RGB-D stream for dense surface reconstruction. To reduce the artifacts from noise and uncertainty of single-view semantic segmentation, we introduce an adaptive integration for fusion and CRF-based semantic label regularization. Experiments demonstrate that our semantic segmentation results of 3D scenes are more precise and detailed compared to the voxel-based previous methods that can only produce low-resolution predictions.

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- *Intrinsic Image Decomposition using Structure-Texture Separation and Surface Normals*: We propose a novel image model for handling textures in intrinsic image decomposition. We also propose a geometric constraint based on surface normals from an RGB-D image to produce locally and globally consistent decomposition results.

- *Texture Map Generation for 3D Reconstructed Scenes*: We propose a novel texture map generation method using spatial and temporal key frames sampling. We also propose an efficient GPU-based texture coordinate optimization method to acquire an accurate texture map.

- *Semantic Reconstruction of 3D Meshes using Volumetric Semantic Fusion*: We propose a novel framework for generating semantically segmented meshes of reconstructed 3D scenes. Proposed volumetric semantic fusion can be seamlessly integrated onto the existing dense 3D reconstruction process to produce precise and detailed semantic segmentation of 3D scenes.

II. Related Work

3D reconstruction of an indoor scene is a classic computer vision problem which researchers have studied over the past decades. As the reconstructed scene can be used in many interactive applications such as virtual and augmented reality, 3D reconstruction has attracted more attention from the industries in recent years. In this chapter, we briefly introduce a brief overview of previous attempts on 3D scene reconstruction in two categories: geometric reconstruction and non-geometric reconstruction.

2.1 Geometric 3D Reconstruction

2.1.1 Reconstruction using Multiple Photographs

Because the single limited field-of-view photograph cannot cover the entire scene at once, 3D reconstruction using multiple captured photographs is a common approach for recovering a large scene. Many researches over a decade [7, 8, 9] on the structure from motion (SfM) and multi-view stereo (MVS) produce the 3D reconstruction of the environment in forms of sparse point clouds by estimating relative camera poses of input photographs and integrating the key points. Based on progress on those SfM algorithms, several researches [10, 3, 11] have been proposed to recover the dense surface geometry of a scene using a patch-based representation.

2.1.2 Real-time Surface Reconstruction

In robotics, simultaneous localization and mapping (SLAM) has been researched for real-time tracking and reconstruction of the scene. SLAM-based methods such as PTAM [12] and DTAM [13] can track a single hand-held RGB

camera in real-time and estimate detailed geometry of the scene. After the consumer-level depth sensors such as Microsoft Kinect are developed, RGB-D sensors have been mainly used to reconstruct the scene geometry. KinectFusion [14, 15] uses a volumetric representation based on truncated signed distance functions (TSDFs) to store the integrated geometry through the reconstruction. Following variants [16, 17] of it focus on improving the limited scalability of the volumetric representation, and achieving globally consistent 3D models in a large-scale [18, 19, 20].

2.2 Non-geometric 3D Reconstruction

As described in the previous chapter, previous 3D reconstruction works mainly focus on the geometric reconstruction rather than recovering the holistic information of the scene. Nevertheless, there have been several attempts to analyze and reconstruct the auxiliary information that represents the environment.

Existing 3D reconstruction methods [14, 20] usually represent the surface color by blending of projected color images during the reconstruction process. It naturally induces blurring and ghosting artifacts in the blended surface colors caused by imprecisely estimated camera poses. Zhou and Koltun [21] proposed an optimization-based method for recovering precise surface color information of the 3D reconstructed scene model.

Because 3D reconstruction only captures the scene lit by a static illumination condition, we need additional lighting condition estimation process to utilize the reconstructed scene for an interactive application such as object composition of augmented reality. Several works have been proposed to automatically estimate the outdoor illumination conditions using a SfM photo collection [22] or a single image [23, 24].

III. Intrinsic Image Decomposition using Structure-Texture Separation and Surface Normals

3.1 Motivation

Intrinsic image decomposition is a problem to decompose an image I into its shading layer S and reflectance layer R based on the following model:

$$I(p) = S(p) \cdot R(p) \tag{3.1}$$

where p is a pixel position. Shading $S(p)$ at p depicts the amount of light reflected at p, and reflectance $R(p)$ depicts the intrinsic color of the material at p, which is invariant to illumination conditions.

Intrinsic image decomposition has been extensively studied in computer vision and graphics communities because it can benefit many computer graphics and vision applications, such as image relighting [25, 26] and material property editing [27]. Since Land and McCann first introduced Retinex algorithm in 1971 [28], various approaches have been introduced for the last a few decades [29, 30, 31]. However, intrinsic image decomposition is still challenging because it is a significantly ill-posed problem where there are two unknowns $S(p)$ and $R(p)$ for one observed data $I(p)$ at each pixel p.

To overcome the ill-posedness, previous methods use constraints, or priors, on shading and reflectance. Shading has been often assumed to be locally smooth, while reflectance assumed to be piecewise constant [28]. While these assumptions work well on simple cases such as Mondrian-like images consisting of patches with constant reflectance, they fail on most natural images. One important charac-

teristic of natural images is their rich textures. Such textures may be due to the reflectance layer (e.g., a flat surface with dotted patterns), or the shading layer (e.g., a surface with bumps and wrinkles causing a shading pattern). Thus, enforcing simple constraints on either or both shading or reflectance layers with no consideration on textures may cause erroneous results.

In this chapter, we propose a novel intrinsic image decomposition model, which explicitly models a separate texture layer T, in addition to the shading layer S and the reflectance layer R. By explicitly modeling textures, S and R in our model depict only textureless base components. As a result, we can avoid ambiguity caused by textures, and use simple constraints on S and R effectively.

To further constrain the problem, we also propose a novel constraint based on surface normal vectors obtained from an RGB-D image. We assume that illumination changes smoothly, and surfaces are Lambertian, i.e., shading of a surface can be determined as a dot product of a surface normal and the light direction. Based on this assumption, our constraint is designed to promote both *local* and *global* consistency of the shading layer based on a locally linear embedding (LLE) framework [32]. For robustness against noise and efficient computation, we sparsely sample points for the surface normal constraint based on local variances of surface normals. This sparse sampling works effectively thanks to our explicit texture modeling. Our shading and reflectance layers do not have any textures, so information from sampled positions can be effectively propagated to their neighbors during the optimization process.

3.2 Related Work

3.2.1 Intrinsic Image Decomposition

Intrinsic image decomposition is a long-standing problem in computer vision. The "Retinex" algorithm was first proposed by Land and McCann in

1971 [28]. The algorithm assumes Mondrian-like images consisting of regions of constant reflectances, where large image gradients are typically caused by reflectance changes, and small gradients are caused by illumination. This algorithm was extended to 2D color images by analyzing derivatives of chromaticity [30].

To overcome the fundamental ill-posedness of the problem, several approaches have been introduced utilizing additional information, such as multiple images [26, 33, 34, 35], depth maps [36, 37, 38], and user interaction [39]. Lee et al. [36] proposed a method, which takes a sequence of RGB-D video frames acquired from a Kinect camera, and proposed constraints on depth information and temporal consistency. In the setting of single input frame, their temporal constraint cannot be used. Barron and Malik [37] proposed joint estimation of a denoised depth map and spatially-varying illumination. However, due to the lack of proper texture handling, textures which should belong to either shading or reflectance layer may appear in the other layer, as shown in Section 3.6.

Recently, Chen and Koltun [38] showed that high quality decomposition results can be obtained by properly constraining shading components using surface normals without joint estimation of a denoised depth map. Specifically, they find a set of nearest neighbors for each pixel based on their spatial positions and normals, and then constrain the shading component of each pixel to be similar to those of its neighbors. While our surface normal based constraint is similar to theirs, there are three different aspects. First, our constraint is derived from the Lambertian surface assumption, which is more physically meaningful. Second, our constraint uses not only spatially close neighbors but also distant neighbors, so we can obtain more globally consistent results. Third, due to our confidence-based sparse sampling, our method can be more efficient and robust to noise.

Another relevant work to ours is Kwatra et al.'s shadow removal approach

for aerial photos [40], which decomposes an aerial image into shadow and texture components. Based on the properties of shadows and textures in aerial images, they define entropy measures for shadows and textures, and minimize them for decomposition. While their work also explicitly considers textures as we do, their work focuses on removal of smooth shadows, such as shadows cast by clouds in aerial images, so it is not suitable for handling complex shadings which are often observed in natural images.

3.2.2 Texture Separation

Structure-texture separation has also been an important topic and extensively studied. Edge-preserving smoothing has been a popular direction, such as anisotropic filtering [41], total variation [42], bilateral filtering [43], nonlocal means filtering [44], weighted least squares filtering [45], and L^0 smoothing [46]. By applying edge-preserving smoothing to an image, small scale textures can be separated from structure components. However, as these approaches rely on local contrasts to distinguish structures and textures, they may fail to properly capture low contrast structures or high contrast textures.

Other approaches have also been proposed to separate textures regardless of their contrasts. Subr et al. [47] estimate envelopes of local minima and maxima, and average the envelopes to capture oscillatory components. Xu et al. [48] proposed a relative total variation measure, which takes account of inherent variation in a local window. Recently, Karacan et al. [49] proposed a simple yet powerful method, which is based on region covariance matrices [50] and nonlocal means filtering [44]. In our work, we adopt Karacan et al.'s approach to separate textures from shading and reflectance components, as it preserves underlying smooth intensity changes.

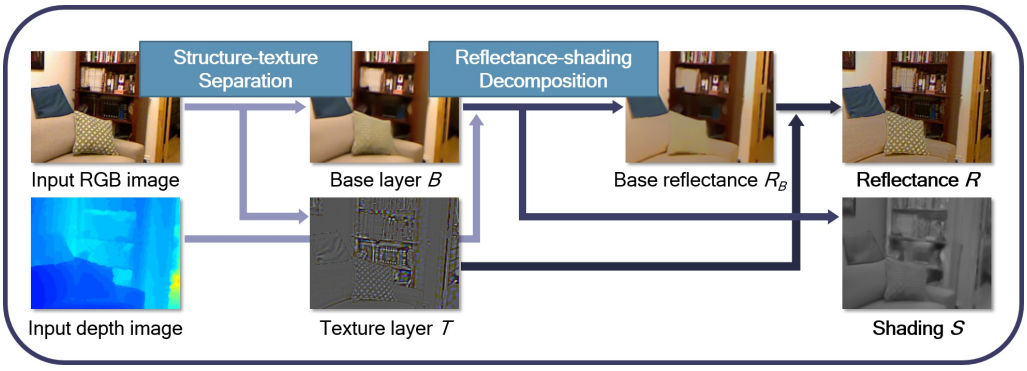


Figure 3.1: Overall process of intrinsic image decomposition framework. First we decompose an input RGB-D image to base and texture layer images, then the base layer image is further decomposed to the base reflectance and shading layers. Finally, we combine the texture layer and base reflectance layer to get the final reflectance image.

3.3 Image Model and Overall Framework

We define a novel model for intrinsic image decomposition as:

$$I(p) = B(p) \cdot T(p) = S_B(p) \cdot R_B(p) \cdot T(p), \tag{3.2}$$

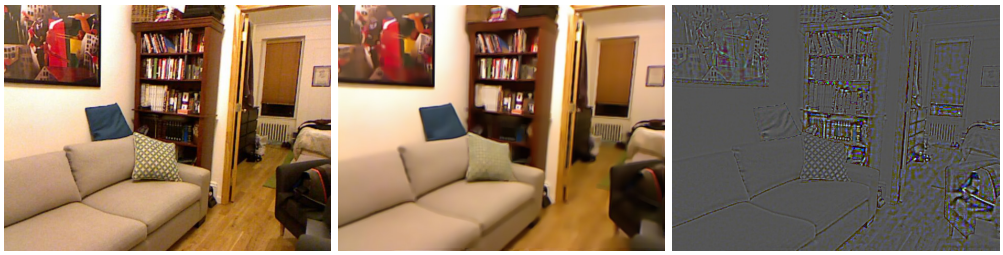
where $B(p) = S_B(p) \cdot R_B(p)$ is a base layer, and $S_B(p), R_B(p)$ and $T(p)$ are shading, reflectance and texture components at a pixel p, respectively. Note that S_B and R_B are different from S and R in Eq. (3.1) as S_B and R_B contain no textures. Based on this model, we can safely assume that R_B is a Mondrian-like image, which is piecewise constant. We also assume that illumination changes smoothly across the entire image, thus S_B is also piecewise smooth with no oscillating variations. Under these assumptions, we will define constraints and energy functions in the following sections, which will be used to decompose an image I into S_B, R_B and T.

As shown in Fig. 3.1, the overall process of our method which consists of two steps can be summarized as follows. In the first step, we decompose an input

RGB image I into a base layer B and a texture layer T. In the second step, the base layer B is further decomposed into a reflectance layer R_B and a shading layer S_B based on the surface normal constraint and other simple constraints. While we use simple constraints similar to previous decomposition methods assuming Mondrian-like images, the constraints can work more effectively as our input for decomposition is B, instead of I, from which textures have been removed. In addition, our global constraint based on surface normals promotes overall consistency of the shading layer, which is hard to achieve by previous methods. Experimental results and comparisons in Section 3.6 demonstrate the effectiveness of our method.

3.4 Decomposition of B and T

In this step, we decompose an input image I into a base layer B and a texture layer T. Texture decomposition has been studied extensively for long time, and several state-of-the-art approaches are available. Among them, we adopt the region covariance based method of Karacan et al. [49], which performs nonlocal means filtering with patch similarity defined using a region covariance descriptor [50]. This method is well-suited for our purpose as it preserves the smoothly varying shading information in the filtering process. We also tested other methods, such as [46, 48], which are based on total variation, but we found that they tend to suppress all small image gradients, including those from shading. Fig. 3.2 shows that the region covariance based method successfully removes textures on the cushion and the floor while preserving shading on the sofa.



(a) Input image (b) Base layer B (c) Texture layer T

Figure 3.2: Decomposition of B and T.

3.5 Decomposition of S and R

After obtaining B, we decompose it into S_B and R_B by minimizing the following energy function:

$$f(S_B, R_B) = f^N(S_B) + \lambda^P f^P(S_B) + \lambda^R f^R(R_B) \tag{3.3}$$

subject to $B(p) = S_B(p) \cdot R_B(p)$. In Eq. (3.3), $f^N(S_B)$ is a surface normal based constraint on S_B, $f^P(S_B)$ is a local propagation constraint on S_B, and $f^R(R_B)$ is a piecewise constant constraint on R_B. In the following, we will describe each constraint in more detail.

3.5.1 Surface Normal based Shading Constraint $f^N(S_B)$

LLE-based local consistency

To derive the surface normal based shading constraint, we first assume that surfaces are Lambertian. On a Lambertian surface, shading S and a surface normal N at p have the following relation:

$$S(p) = i_L \cdot \langle L(p), N(p) \rangle \,, \tag{3.4}$$

where i_L is an unknown light intensity, $L(p)$ is the lighting direction vector at p, and $\langle L(p), N(p) \rangle$ is the inner product of $L(p)$ and $N(p)$.

As we assume that illumination changes smoothly, we can also assume that i_L and L are constant in a local window. We can express a surface normal at

p as a linear combination of normals at its neighboring pixels $q \in \mathcal{N}_l(p)$, i.e. $N(p) = \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N N(q)$ where w_{pq}^N is a linear combination weight. Then, $S(p)$ can also be expressed as the same linear combination of the neighbors $S(q)$:

$$S(p) = i_L \cdot \left\langle L, \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N N(q) \right\rangle = \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N \left(i_L \cdot \langle L, N(q) \rangle \right) = \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N S(q). \tag{3.5}$$

Based on this relation, we can define a local consistency constraint $f_l^N(S_B)$ as:

$$f_l^N(S_B) = \sum_{p \in \mathcal{P}_N} \left(S_B(p) - \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N S_B(q) \right)^2, \tag{3.6}$$

where $\mathcal{P}_N$ is a set of pixels. Note that we could derive this constraint without having to know the value of the light intensity i_L.

Interestingly, Eq. (3.5) can be interpreted as a LLE representation [32]. LLE is a data representation, which projects a data point from a high dimensional space onto a low dimensional space by representing it as a linear combination of its neighbors in the feature space. Adopting the LLE approach, we can calculate the linear combination weights $\{w_{pq}^N\}$ by solving:

$$\underset{\{w_{pq}^N\}}{\operatorname{argmin}} \sum_{p \in \mathcal{P}_N} \| N(p) - \sum_{q \in \mathcal{N}_l(p)} w_{pq}^N N(q) \|^2, \tag{3.7}$$

subject to $\sum_{q \in \mathcal{N}_l(p)} w_{pq}^N = 1$.

To find $\mathcal{N}_l(p)$, we build a 6D vector for each pixel as $[x(p), y(p), z(p), n_x(p), n_y(p), n_z(p)]^T$, where $[x(p), y(p), z(p)]^T$ is the 3D spatial location obtained from the input depth image, and $[n_x(p), n_y(p), n_z(p)]^T$ is the surface normal at p. Then, we find the k-nearest neighbors using the Euclidean distance between the feature vectors at p and other pixels.

Global consistency

While a locally consistent shading result can be obtained with $f_l^N(S_B)$, the result may be still globally inconsistent. Imagine that we have two flat regions,

which are close to each other, but their depths are slightly different. Then, for each pixel in one region, all of its nearest neighbors will be found in the same region, and the two regions may end up with completely different shading values. This phenomenon can be found in Chen and Koltun's results in Fig. 3.7, as their method promotes only local consistency. In their shading result on the first row, even though the cushion on the sofa should have similar shading to the sofa and the wall, they have totally different shading values.

In order to avoid such global inconsistency, we define another constraint $f_g^N(S_B)$, which promotes global consistency:

$$f_g^N(S_B) = \sum_{p \in \mathcal{P}_N} \left(S_B(p) - \sum_{q \in \mathcal{N}_g(p)} w_{pq}^N S_B(q) \right)^2, \tag{3.8}$$

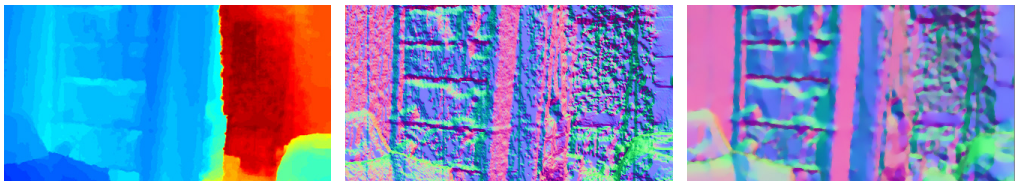
where $\mathcal{N}_g(p)$ is a set of global k-nearest neighbors for each pixel p. To find $\mathcal{N}_g(p)$, we simply measure the Euclidean distance between the surface normals at p and other pixels without considering their spatial locations, so that the resulting $\mathcal{N}_g(p)$ can include spatially distant pixels. With the two constraints, we define the constraint $f^N(S_B)$ as:

$$f^N(S_B) = f_l^N(S_B) + \lambda_g^N f_g^N(S_B), \tag{3.9}$$

where λ_g^N is the relative weight for f_g^N. We set $k = 20$ for both local and global consistency constraints.

Sub-sampling for efficiency and noise handling

It can be time-consuming to find k-nearest neighbors and apply $f^N(S_B)$ for every pixel in an image. Moreover, depth images from commercial RGB-D cameras are often severely noisy as shown in Fig. 3.3a. We may apply a recent depth map denoising method, but there can still remain significant errors around depth discontinuities causing a noisy normal map (Fig. 3.3c).



(a) Raw depth image (b) Raw normal map (c) Denoised normal map

Figure 3.3: Depth images from commercial RGB-D cameras often suffer from severe noise, which is difficult to remove using a denoising method.

To improve the efficiency and avoid noisy normals, we propose a sub-sampling based strategy for building $\mathcal{P}_N$. Specifically, we divide an image into a uniform grid. In each grid cell, we measure the variance of the surface normals in a local window centered at each pixel. Then, we choose a pixel with the smallest variance. This is because complex scene geometry is more likely to cause severe depth noise, so we would better choose points in a smooth region with low variance. We also find the nearest neighbors for $\mathcal{N}_l(p)$ and $\mathcal{N}_g(p)$ from $\mathcal{P}_N$ to avoid noisy normals and accelerate the nearest neighbor search. While we use the constraint $f^N(S_B)$ only for sub-sampled pixel positions, information on the sampled positions can be propagated to neighboring pixels during the optimization due to the constraint $f^P(S_B)$, which is described next.

3.5.2 Local Propagation Constraint $f^P(S_B)$

Since we use subsampled pixel positions for the constraint $f^N(S_B)$, other pixels do not have any shading constraint. To properly constrain such pixels, we propagate the effects of $f^N(S_B)$ to neighboring pixels using two local smoothness constraints on shading. Specifically, we define $f^P(S_B)$ as:

$$f^P(S_B) = f^P_{lap}(S_B) + \lambda^P_N f^P_N(S_B), \tag{3.10}$$

where $f^P_{lap}(S_B)$ is based on the structure of the base layer B, and $f^P_N(S_B)$ is based on surface normals.

Since all the textures are already separated out to T and we assume that R_B is piecewise constant, we can safely assume that small image derivatives in B are from the shading layer S_B. Then, S_B can be approximated in a small local window as:

$$S_B(p) \approx aB(p) + b, \tag{3.11}$$

where $a = \frac{1}{B_f - B_b}$ and $b = \frac{-B_b}{B_f - B_b}$, and B_f and B_b are two primary colors in the window. This approximation inspires us to use the matting Laplacian [51] to propagate information from the sub-sampled pixels to their neighbors in a structure-aware manner. Specifically, we define the first constraint for propagation using the matting Laplacian as follows:

$$f_{lap}^{P}(S_B) = \sum_{k} \sum_{(i,j)\in\omega_k} w_{ij}^{lap}(S_B(i) - S_B(j))^2, \tag{3.12}$$

where ω_k is the k-th local window. w_{ij}^{lap} is the (i,j)-th matting Laplacian element, which is computed as:

$$w_{ij}^{lap} = \sum_{k|(i,j)\in\omega_k} \left\{ \delta_{ij} - \frac{1}{|\omega_k|} \left(1 + \left(B(i) - \mu_k^B\right) \left(\Sigma_k^B + \frac{\epsilon}{|\omega_k|} I_3 \right)^{-1} \left(B(j) - \mu_k^B\right) \right) \right\}, \tag{3.13}$$

where ϵ is a regularizing parameter, and $|\omega_k|$ is the number of pixels in the window ω_k. δ_{ij} is Kronecker delta. μ_k^B and Σ_k^B are the mean vector and covariance matrix of B in ω_k, respectively. I_3 is a 3×3-identity matrix.

This constraint is based on the key advantage of removing textures from the input image. With the original image I, because of textures, information at sample points cannot be propagated properly while being blocked by edges introduced by textures. In contrast, by removing textures from the image, we can effectively propagate shading information using the structure of the base image B, obtaining higher quality shading results.

The second local smoothness constraint $f_N^P(S_B)$ is based on surface normals. Even if surface normals are noisy, they still provide meaningful geometry infor-

mation for smooth surfaces. Thus, we formulate a constraint to promote local smoothness based on the differences between adjacent surface normals as follows:

$$f_N^P(S_B) = \sum_{p\in\mathcal{P}} \sum_{q\in\mathcal{N}(p)} w_{pq}^N \left(S_B(p) - S_B(q)\right)^2, \tag{3.14}$$

where $\mathcal{P}$ is a set of all pixels in the image and $\mathcal{N}(p)$ is the set of 8-neighbors of p. w_{pq}^N is a continuous weight, which is defined using the angular distance between normal vectors at p and q:

$$w_{pq}^N = \exp\left(-\frac{1 - \langle N(p), N(q)\rangle^2}{\sigma_n^2}\right), \tag{3.15}$$

where we set $\sigma_n = 0.5$. w_{pq}^N becomes close to 1 if the normals $N(p)$ and $N(q)$ are similar to each other, and becomes small if they are different.

Fig. 3.4 shows the effect of the constraint $f^P(S_B)$. The local propagation constraint enables shading information obtained from the LLE-based constraints to be propagated to other pixels, which results in clear shading near edges.

3.5.3 Reflectance Constraint $f^R(R_B)$

The constraint $f^R(R_B)$ is based on a simple assumption, which is used by many Retinex-based approaches [30, 36, 52, 53]: if two neighboring pixels have the same chromaticity, their reflectance should be the same as well. Based on this assumption, previous methods often use a constraint defined as:

$$f^R(R_B) = \sum_{p\in\mathcal{P}} \sum_{q\in\mathcal{N}(p)} w_{pq}^R \left(R_B(p) - R_B(q)\right)^2. \tag{3.16}$$

The weighting function w_{pq}^R is a positive constant if the difference between the chromaticity at p and q is smaller than a certain threshold, and zero otherwise. For this constraint, it is critical to use a good threshold, but finding a good threshold is non-trivial and can be time consuming [53].

Instead of using a threshold, we use a continuous weight $w_{pq}^{R'}$, which involves chromaticity difference between two pixels p and q in the form of angular distance

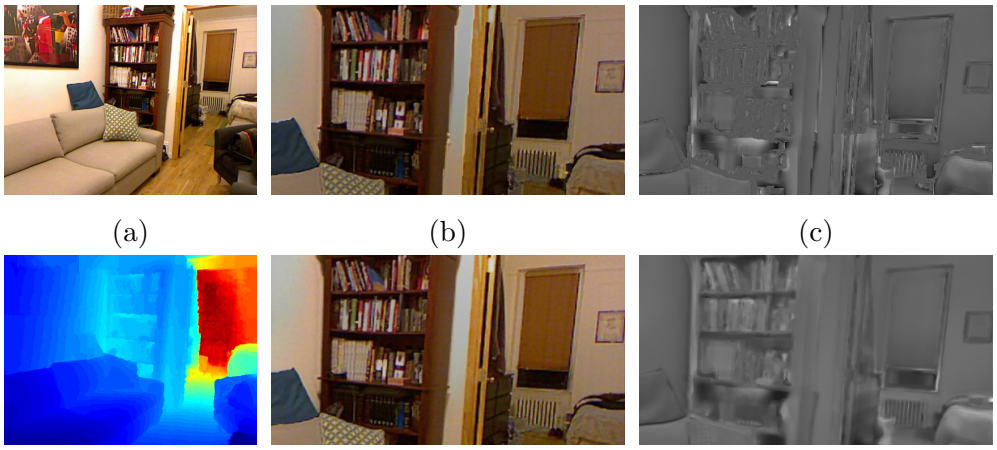


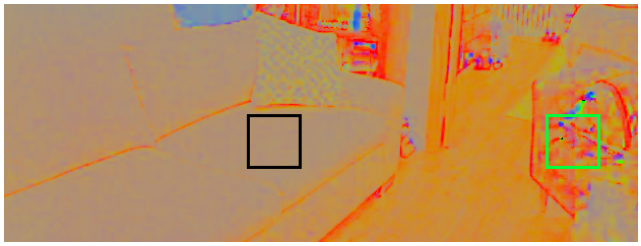
Figure 3.4: Effect of the local propagation constraint $f^P(S_B)$. (a,d) Input RGB and depth images. (b-c) Reflectance and shading results without $f^P(S_B)$. (e-f) Reflectance and shading results with $f^P(S_B)$. Without the constraint, shading of pixels which have not been sampled for normal based constraint $f^N(S_B)$ are solely determined by the reflectance constraint through the optimization process. As a result, (c) shows artifacts on the bookshelves due to the inaccurate reflectance values.

between two directional vectors:

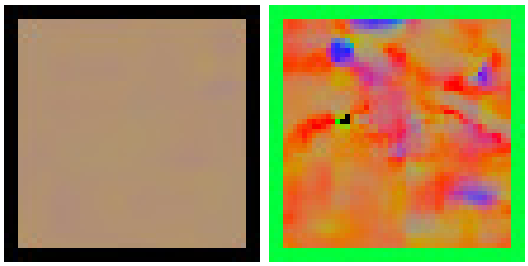
$$w_{pq}^{R'} = \exp\left(-\frac{1-\langle C(p), C(q)\rangle^2}{\sigma_c^2}\right)\left\{1+\exp\left(-\frac{B(p)^2+B(q)^2}{\sigma_i^2}\right)\right\}, \qquad (3.17)$$

where chromaticity $C(p) = B(p)/|B(p)|$ is a normalized 3D vector consisting of RGB color channels. We set $\sigma_c^2 = 0.0001, \sigma_i^2 = 0.8$. The first exponential term measures similarity between $C(p)$ and $C(q)$ using their angular distance. The term becomes 1 if $C(p) = C(q)$ and becomes close to 0 if $C(p)$ and $C(q)$ are different from each other, so that consistency between neighboring pixels can be promoted only when $C(p)$ and $C(q)$ are close to each other. The second exponential term is a darkness weight. Due to the nature of imaging sensors, dark pixels suffer from noise more than bright pixels, causing severe chromaticity noise in dark regions such as shadows (Fig. 3.5). Thus, the second term gives larger weights to dark pixels to overcome such chromaticity noise. Using the

(a)



(b)



(c)

Figure 3.5: Comparison of chromaticity noise between a bright region (black box) and a dark region (green box). (a) Input image. (b) Chromaticity of the input image. (c) Magnified views of two regions.

weight $w_{pq}^{R'}$, our constraint $f^R(R_B)$ is defined as:

$$f^R(R_B) = \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{N}(p)} w_{pq}^{R'} \left(R_B(p) - R_B(q)\right)^2 . \tag{3.18}$$

3.5.4 Optimization

To simplify the optimization, we take the logarithm to our model as done in [53, 36, 38]. Then, we get $\log B(p) = s_B(p) + r_B(p)$ where $s_B(p) = \log S_B(p)$ and $r_B(p) = \log R_B(p)$. We empirically found that proposed constraints could also represent the similarities between pixels in the logarithmic domain, even for the

LLE weights. Thus, we approximate the original energy function (Eq. (3.3)) as:

$$f(s_B) = f^N(s_B) + \lambda^P f^P(s_B) + \lambda^R f^R(\log B(p) - s_B(p)). \tag{3.19}$$

As all the constraints f^N, f^P, and f^R are quadratic functions, Eq. (3.19) is a quadratic function of s_B. We minimize Eq. (3.19) using a conjugate gradient method. We used $\lambda^P = \lambda^R = 4, \lambda_g^N = 1$, and $\lambda_N^P = 0.00625$.

3.6 Experimental Results

3.6.1 Experimental Setting

For evaluation, we implemented our method using Matlab, and used the structure-texture separation code of the authors [49] for decomposition of B and T. For a 624×468 image, decomposition of B and T takes about 210 seconds, and decomposition of S and R takes about 150 seconds on a PC with Intel Core i7 2.67GHz CPU, 12GB RAM, and Microsoft Windows 7 64bit OS.

For evaluation, we selected 14 images from the NYU dataset [54], which provides 1,400 RGB-D images. When selecting images, we avoided images which do not fit our Lambertian surface assumption, e.g., images with glossy surfaces such as a mirror or a glass. Fig. 3.6 shows some of the selected images. It is worth mentioning that, while Chen and Koltun [38] used the synthetic dataset of [55] to quantitatively evaluate their approach, in our evaluation, we did not include such quantitative evaluation. This is because the synthetic dataset of [55] was not generated for the purpose of intrinsic image decomposition benchmark, so its ground truth reflectance and shading images are not physically correct in many cases, such as shading of messy hairs and global illumination.

In our image model $I(p) = S_B(p) \cdot R_B(p) \cdot T(p)$, texture T can contain not only reflectance textures, but also shading textures caused by subtle and complex geometry changes, which are often not captured in a noisy depth map. In our

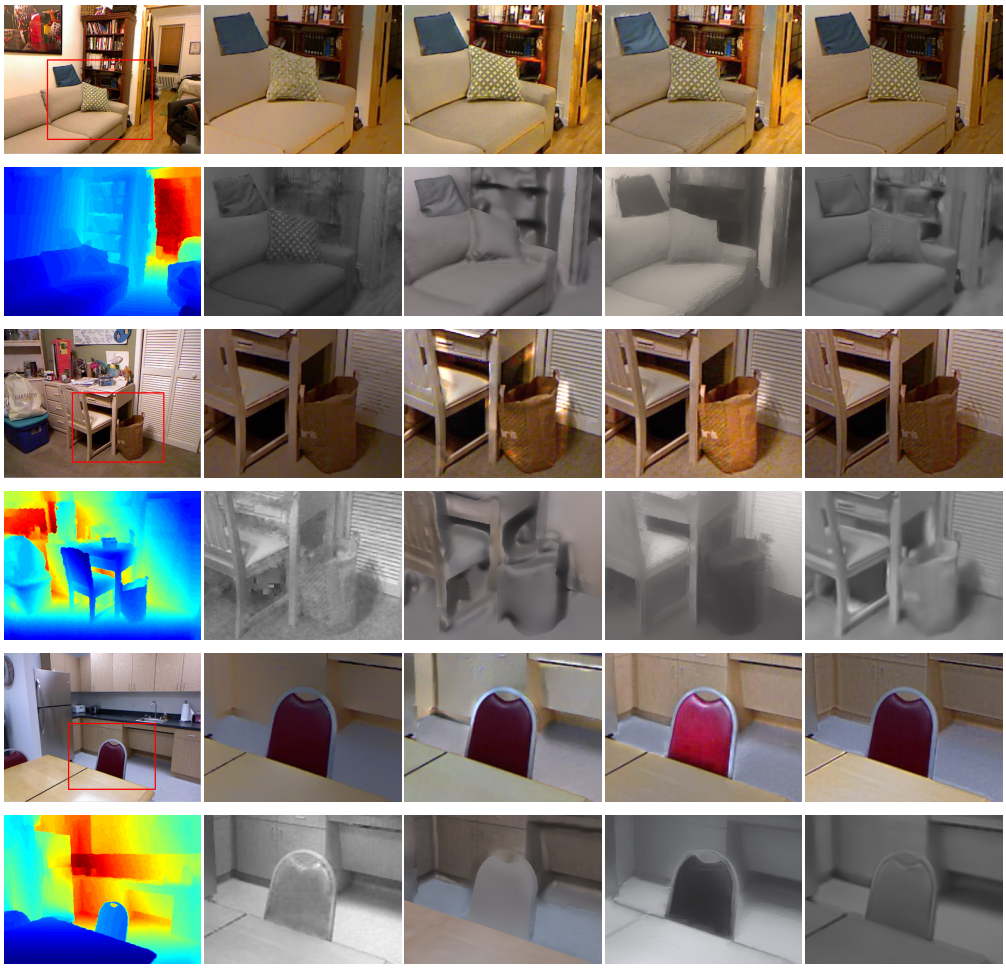
Figure 3.6: Test images from the NYU dataset [54].

work, we do not further decompose T into reflectance and shading textures. Instead, for visualization and comparison, we consider T as a part of the reflectance layer R, i.e., $R(p) = R_B(p) \cdot T(p)$ and $S(p) = S_B(p)$. That is, every reflectance result in this chapter is the product of the base reflectance layer R_B and the texture layer T.

3.6.2 Qualitative Evaluation

We evaluated our method using qualitative comparisons with other methods. Fig. 3.7 shows results of three other methods and ours. The conventional color Retinex algorithm [56] takes a single RGB image as an input. This method highly depends on its reflectance smoothness constraint, which can be easily discouraged by rich textures. Thus, its shading results contain a significant amount of textures, which should be in the reflectance layers (e.g., patterned cushion in the sofa scene).

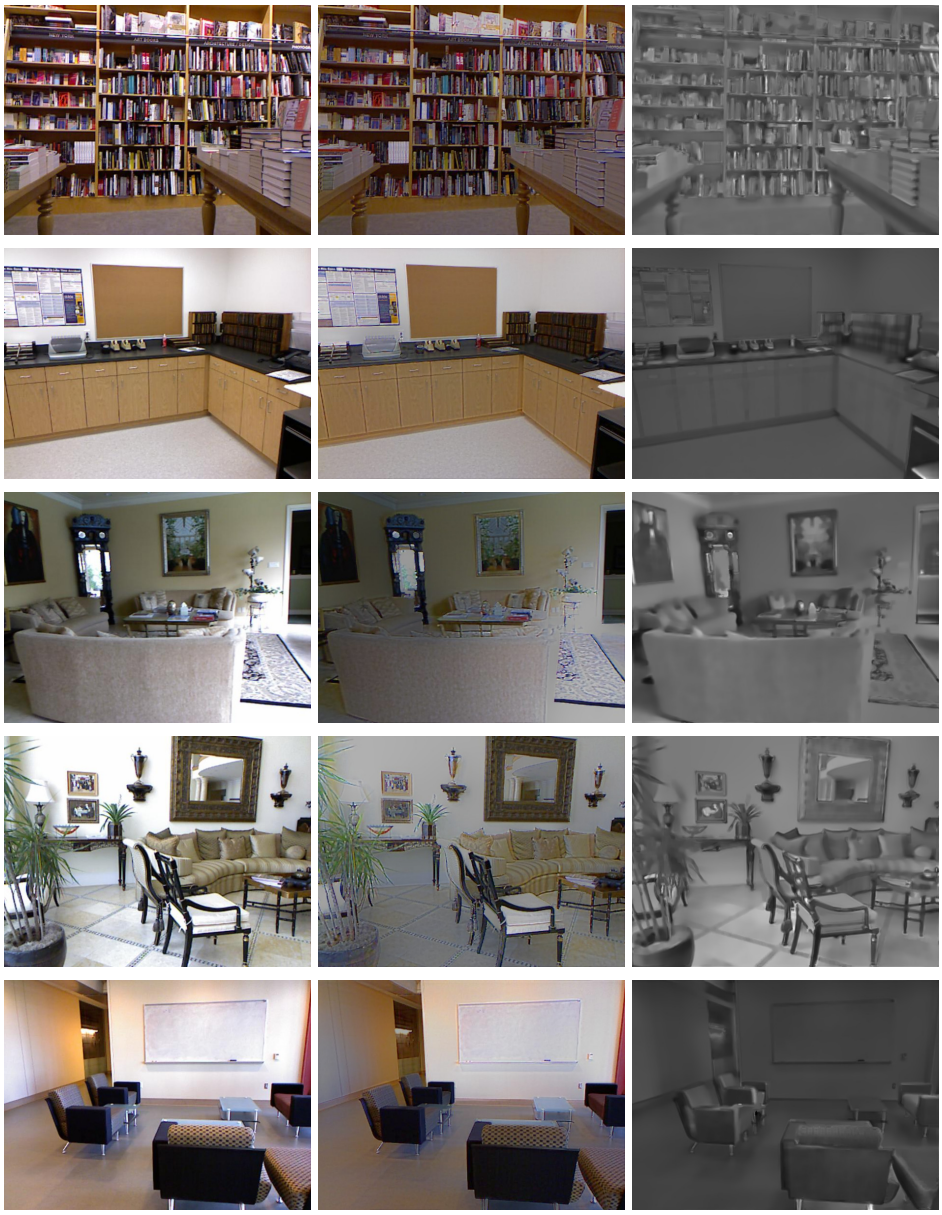
Barron and Malik [37] jointly estimate a refined depth map and spatially varying illumination, and obtain a shading image from that information. Thus, their shading results in Fig. 3.7 do not contain any textures on them. However, their shading results are over-smoothed on object boundaries because of their incorrect depth refinement (e.g., the chair and bucket in the desk scene).



(a) (b) (c) (d) (e)

Figure 3.7: Decomposition result comparison with other methods. (a) Input. (b) Retinex [56]. (c) Barron and Malik [37]. (d) Chen and Koltun [38]. (e) Ours.

The results of Chen and Koltun [38] show more reliable shading results than [56, 37], even though they do not use any explicit modeling for textures. This is because of their effective shading constraints based on depth cues. However, as mentioned in Section 3.2, due to the lack of global consistency constraints, their shading results often suffer from the global inconsistency problem (e.g., the chair in the kitchen scene). On the contrary, our method produces well-decomposed textures (e.g., cushion in the sofa scene) and globally consistent shading (e.g., the chair and the bucket in the desk scene) compared to the other three methods.



input image reflectance shading

Figure 3.8: Additional intrinsic decomposition results of NYU-D dataset.

Additional decomposition results of our method are shown in Fig. 3.8.

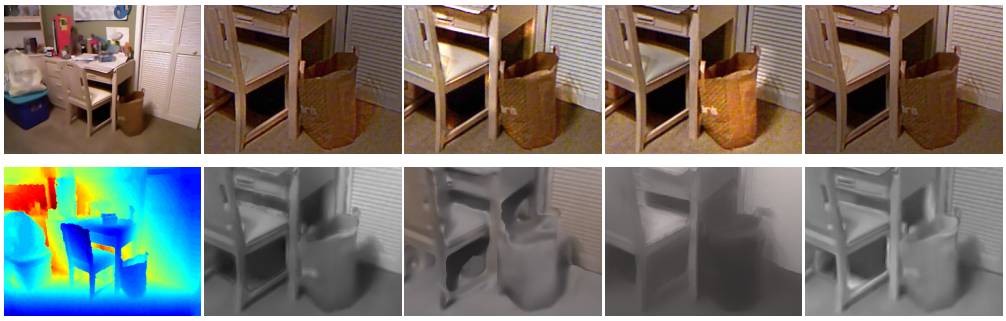
(a) (b) (c)

Figure 3.9: Decomposition results without and with the texture filtering step. (a) Input image. (b) Reflectance and shading results without texture filtering step. (c) Reflectance and shading results with texture filtering step.

3.6.3 Analysis on Texture Filtering

To show the effect of our texture filtering step, we also tested our algorithm without texture filtering (Fig. 3.9). Thanks to our non-local shading constraints, the shading results are still globally consistent (Fig. 3.9b). However, the input image without texture filtering breaks the Mondrian-like image assumption, so lots of reflectance textures remain in the shading result. This experiment shows that our method fully exploits properties of the texture filtered base image, such as piecewise-constant reflectance and texture-free structure information.

We also conducted another experiment to clarify the effectiveness of our decomposition step. This time, we fed texture-filtered images to previous methods as their inputs. Fig. 3.10 shows texture filtering provides some improvements to other methods too, but the improvements are not as big as ours. Retinex [56] benefited from texture filtering, but the method has no shading constraints and the result still shows globally inconsistent shading (the bucket and the closet). Big improvements did not happen with recent approaches [37, 38] either. [37] strongly uses its smooth shape prior, which causes over-smoothed shapes and



(a) (b) (c) (d) (e)

Figure 3.10: Decomposition results of other methods using a texture-filtered input. (a) Input base image. (b) Retinex [56]. (c) Barron and Malik [37]. (d) Chen and Koltun [38]. (e) Ours.

shading in regions with complex geometry. In the result of [38], globally inconsistent shading still remains due to the lack of global consistency constraints.

3.7 Applications

One straightforward application of intrinsic image decomposition is material property editing such as texture composition. Composing textures into an image naturally is tricky, because it requires careful consideration of spatially varying illumination. If illumination changes such as shadows are not properly handled, composition results may look too flat and artificial (Fig. 3.11b). Instead, we can first decompose an image into shading and reflectance layers, and compose new textures into the reflectance layer. Then, by recombining the shading and reflectance layers, we can obtain a more naturally-looking result (Fig. 3.11c).

Another application is image relighting (Fig. 3.12). Given an RGB-D input image, we can generate a new shading layer using the geometry information obtained from the depth information. Then, by combining the new shading layer with the reflectance layer of the input image, we can produce a relighted image.

(a) Original image

(b) Naive copy & paste

(c) Our method

Figure 3.11: Texture composition.

3.8 Discussion

Although intrinsic image decomposition has been extensively studied in computer vision and graphics, the progress has been limited by the nature of natural images, especially rich textures. In this chapter, we proposed a novel image model, which explicitly models textures for intrinsic image decomposition. With

Original images Relighted images

Figure 3.12: Image relighting using decomposed results

explicit texture modeling, we can avoid confusion on the smoothness property caused by textures and can use simple constraints on shading and reflectance components. To further constrain the decomposition problem, we additionally proposed a novel constraint based on surface normals obtained from an RGB-D image. Assuming Lambertian surfaces, we formulated our surface normal based constraints using a LLE framework [32] in order to promote both local and global consistency of shading components.

In our experiments, we assumed textures to be a part of reflectance for the purpose of comparison with other methods. However, textures may be caused by either or both of reflectance and shading, as we mentioned in Introduction. As future work, we plan to further decompose textures into reflectance and shading texture layers using additional information such as surface geometry.

IV. Texture Map Generation for 3D Reconstructed Scenes

4.1 Motivation

Nowadays RGB-D cameras, such as Microsoft Kinect, have become widely available. Various researches on 3D reconstruction based on RGB-D images, e.g., KinectFusion and its variations [4, 6, 57, 58], enabled 3D navigation of a scene by rendering the reconstructed 3D model from desirable viewpoints. However, these reconstructed 3D models are not yet popularly used in applications, such as virtual reality and augmented reality, due to the lack of accurate color information. Most reconstruction methods so far recover the color information by blending the corresponding color values in different input images. Imprecisely estimated camera poses and lens distortions can cause misalignments between images, and consequently, this simple blending may induce blurring and ghosting artifacts in the blended surface colors and diminish the quality of the rendering results.

Recently Zhou and Koltun [21] proposed an optimization-based approach for mapping color images onto a 3D geometric model reconstructed using a consumer RGB-D sensor. They optimized the camera poses for all color images and applied non-rigid transforms to color images to enhance the alignments with the 3D model. In their approach, the 3D reconstructed model is rendered using the *vertex colors* computed by weighted averages of aligned color values. To represent the detailed color information of a large reconstructed scene, it would require a large number of vertices that introduces storage and rendering overheads. In addition, their alternating optimization takes several minutes for a small object, and the approach is not much scalable to handle large indoor scenes.

In this chapter, we propose a novel approach based on *texture mapping* for mapping color images onto a 3D reconstructed scene. Differently from previous methods [6, 21] that use voxel or vertex colors to render 3D reconstructed models, our approach uses an accurately recovered texture map for rendering. The usage of texture mapping enables a simpler mesh to be used for reconstructing detailed color information of a large scene, such as a whole room.

However, it is not straightforward to generate a texture map without blurring and ghosting artifacts from input color images, as the color images and the reconstructed 3D model should be precisely aligned to produce such a texture map. Although it could be possible to generate a texture map from vertex colors, such an approach would need optimized colors for a huge number of vertices to handle a large scene. To efficiently generate an accurate texture map usable for rendering a large 3D reconstructed model, we present a novel framework for mapping color images onto a 3D model with the following technical contributions:

- texture map generation method that maximizes the photometric consistency of input images in the global texture map.
- efficient optimization of the texture map generation method based on a parallel Gauss-Newton solver running on GPU.
- spatio-temporal adaptive sampling of input images that reduces the redundant information and motion blurs for faster processing and sharper texture maps.

Experimental results on various RGB-D image data demonstrate that the proposed method reconstructs high-quality texture maps within a few seconds for small objects and within practical running times for large indoor scenes.

4.2 Related Work

To generate 3D models for real objects or scenes from given images, 3D reconstruction has been widely researched in computer graphics and vision [59, 60, 61]. As depth cameras have become popular, recent developments focus on reconstruction using RGB-D images. *KinectFusion* proposed by Newcombe et al. [4] is one of the pioneering works that reconstructs the 3D geometry using a volumetric representation. Based on *KinectFusion*, several extensions [4, 6, 58, 5] have been proposed. Among them, Nießner et al. [6] proposed a large-scale indoor scene reconstruction method using a hash-based geometric representation. In our approach, we use their method in the model reconstruction step to generate a 3D mesh with estimation of the camera poses of input color images. We also use the 3D models and corresponding camera poses reconstructed by [5] for the experiments.

Beyond the 3D geometry reconstruction, mapping the color information from given multiple images onto the reconstructed model has not been heavily investigated yet. In most 3D reconstruction methods [4, 6], estimated relative camera poses between input images are used to average the pixel colors for object points. Hence, imprecise alignments between color images cause blurring and ghosting artifacts, which lower the rendering quality. Our texture coordinate optimization resolves the misalignments by imposing photometric consistency among color images projected on the reconstructed model.

The color map optimization method proposed by Zhou and Koltun [21] is one of the state-of-the-art color reconstruction techniques. After reconstructing a 3D model from a depth image stream, they optimize the alignments among color images with respect to every vertex in the model. They also optimize a non-rigid deformation function for each image to correct distortions due to imprecise 3D reconstruction. Although their method can precisely refine the color mapping for

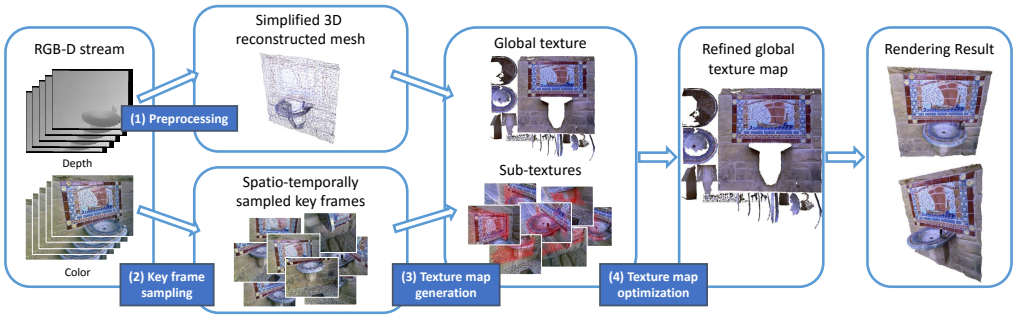


Figure 4.1: Overall process of our texture map generation method for a 3D reconstructed scene. Input color images are mapped and blended on a simplified version of the 3D reconstructed mesh to produce a global texture map. Our method refines the global texture map by optimizing the texture coordinates of multiple sub-textures mapped onto each face of the simplified 3D mesh.

a reconstructed 3D object, it takes several hundreds of seconds for a small object due to its time-consuming non-linear optimization. By applying texture mapping to a 3D model and exploiting the locality of texture coordinate optimization, our method can generate a high-quality texture map within a few seconds for a small object.

4.3 Overview

4.3.1 Overall process

Fig. 4.1 shows the overall process of our approach to generate a texture map for a reconstructed 3D model. We use a depth and color image stream as the input. For a given input stream, we reconstruct a geometric model using a voxel-based 3D reconstruction method, and the reconstructed model is simplified using a mesh simplification method. We then select key frames from the color image stream using spatio-temporal adaptive sampling to reduce the processing time and to increase the quality of the generated texture map.

To generate a global texture map, we first estimate multiple sub-textures

(a) (b)

Figure 4.2: Mesh simplification example. The number of faces is reduced from (a) 460K to (b) 23K.

for each face by projecting its vertices onto the selected key frames in which the face is visible. The global texture map for a face is determined by weighted blending of the multiple sub-textures. The initial sub-textures can be photometrically inconsistent to each other because of imprecisely estimated camera poses and possible distortions in the key frames. We refine the sub-textures of faces by optimizing an objective function of sub-texture coordinates to maximize the photometric consistency. This optimization enables us to obtain a precise global texture map that can be rendered efficiently with the simplified mesh.

4.3.2 Preprocessing of input data

Geometric model reconstruction For a given depth and color image stream, we first reconstruct a 3D geometric model with a triangular mesh representation. Any 3D reconstruction method can be used for this step. In our work, we use [6, 5] to obtain a 3D model from a given RGB-D image stream. For [5], we used the reconstructed 3D models provided by the authors for experiments.

Model simplification Even for a small scene, the initial mesh $\mathbf{M^0}$ usually consists of millions of vertices and faces, which are too heavy for the remaining process. In addition, rendering with texture mapping becomes inefficient when the model contains too large number of faces. To address these issues, we apply a mesh simplification method with quadric error metric [62] to $\mathbf{M^0}$ for reducing the number of faces while preserving the shape. In Fig. 4.2, we reduced the number of faces significantly, up to 5% of the original, but geometric details are still almost preserved. The simplified mesh is denoted by $\mathbf{M}$, which is the input of the remaining process.

4.4 Spatio-temporal Key Frame Sampling

The length of an input RGB-D stream can vary from several hundreds to thousands of frames, according to the scale of the target scene to be reconstructed. Such a long stream contains a lot of redundant data, most of which are less useful for texture map generation. Also, the color image stream captured by a hand-held camera suffers from motion blurs that would lower the quality of the generated texture map. By sampling the input images according to the uniqueness and quality, we can accelerate the texture generation process and obtain a better quality texture map. We call this key frame selection process *spatio-temporal sampling*.

4.4.1 Temporal sampling using blurriness

We first sample the images in the temporal domain by selecting relatively less blurred frames. We use the method of Crete et al. [63] to measure the blurriness of input frames. Similarly to Zhou and Koltun [21], we select key frames one by one with the smallest blurriness values. Specifically, we first check the blurriness of σ_{max} frames from the beginning of the stream and choose the key frame with

the smallest blurriness. We then skip σ_{min} frames from the last selected key frame to avoid redundant frames. Then again, we check σ_{max} frames from the last skipped frame to select the next key frame, and repeat the process. We used $\sigma_{min} = 5$ and $\sigma_{max} = 30$ for all experiments in our work.

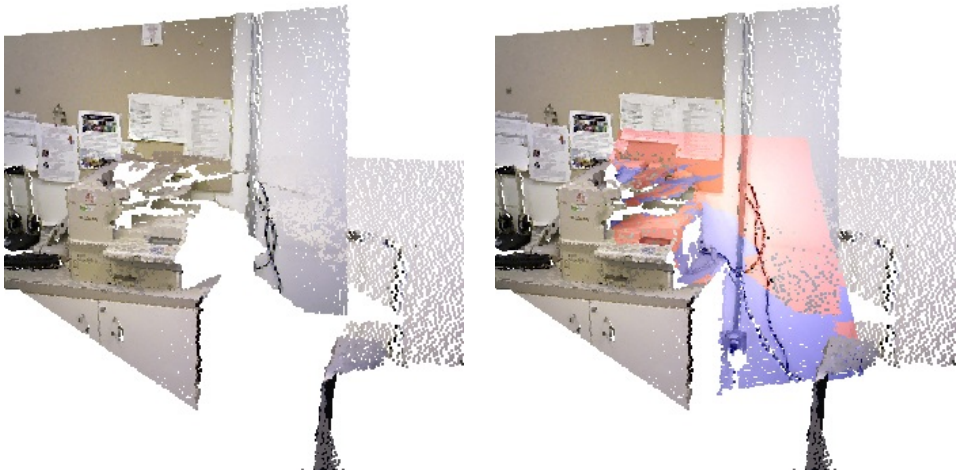
Note that we use a shorter time interval (several frames) for key frame selection than [21] which uses one to five seconds for the interval. Our method aims to handle reconstruction of indoor scenes rather than small objects. In a stream capturing an indoor scene with clutters, small parts of the scene could be captured only in small portions of the stream. If a long time interval is used for key frame selection, we can easily miss the details of the scene. Therefore we use a shorter time interval to select high quality key frames as many as possible, and filter out the redundant frames using spatial sampling.

4.4.2 Spatial sampling using uniqueness

After sampling key frames in the temporal domain with blurriness, we perform sampling in the spatial domain. To reduce redundant information, we sample key frames that contain unique information which is distinguishable from others. To measure the uniqueness of a frame I, we first define the uniqueness of a pixel p in I against a set of temporally sampled key frames $\mathbf{I}_t$ as:

$$q_I(p) = \begin{cases} 1, & \text{if } |z_I(p) - z_{I'}(p')| > \sigma_s \ \forall I' \in \mathbf{I}_t, \\ 0, & \text{otherwise,} \end{cases} \tag{4.1}$$

where p' is the 2D position when p in I has been projected onto I', $z_I(p)$ is the depth value of pixel p in I, and σ_s is a user-specified overlapping threshold. That is, a depth pixel is unique when it does not overlap with any other depth pixels in $\mathbf{I}_t$. Fig. 4.3 shows examples of unique and overlapping depth pixels. Based on



(a) (b)

Figure 4.3: Frame uniqueness measurement. (a) temporally sampled key frames $\mathbf{I}_t$, (b) unique regions (blue points) and overlapping regions (red points) of a frame I against the frame set $\mathbf{I}_t$.

the pixel uniqueness, the uniqueness of frame I is evaluated as:

$$Q(I) = \frac{1}{|I|} \sum_{p \in I} q_I(p), \tag{4.2}$$

where $|I|$ is the number of pixels in I.

Evaluating the uniqueness for all temporally sampled key frames would be time consuming, as we should project every pixel in each key frame onto all other key frames. As the number of key frames increases, computation time for evaluating the uniqueness increases accordingly. For a large-scale scene that consists of thousands of images, it may take few hours. To reduce the computation time, we can use two types of acceleration: downsampling and parallelization.

We can approximate the frame uniqueness by downsampling an image into uniform grids and only using the uniqueness of representative points of the grids. We use the average of all depth pixels in each grid as the representative point. In our implementation, we use 10×10 uniform grids. Since each grid can cover a large region of the scene, we use the overlapping threshold $\sigma_s = 20\ cm$ for all

full image sampling downsampling

Figure 4.4: Results with/without frame uniqueness approximation. The quality of the final rendering result is not much affected.

results in this chapter. While preserving the quality, this approximation gives huge acceleration compared to full image processing. Fig. 4.4 shows that the quality of the final rendering result is not much affected by this approximation. Another option for acceleration is to use GPU for parallel implementation of the full image sampling. Our CUDA implementation of the frame uniqueness computation based on full images runs twice faster than the CPU-based downsampling version.

After calculating the uniqueness of all key frames, we remove the key frames with the smallest uniqueness values one by one while updating the uniqueness of neighboring key frames after each removal. Two images are neighbors when they have overlapped representative points. The removal process continues until the smallest uniqueness becomes large enough, resulting in key frames that have enough unique information among each other. This spatial sampling of key frames can be efficiently performed by maintaining a priority queue for the uniqueness values.

As a result of temporal and spatial sampling, the size of the input stream

is significantly reduced from thousands to tens or hundreds of frames. Despite of this aggressive sampling, the quality of final rendering results are not much degraded, as demonstrated in Fig. 4.10.

4.5 Texture Map Generation

After we have chosen a set of key frames using spatio-temporal sampling, we use them to generate a global texture map for the simplified 3D mesh. This process consists of two steps, global texture map generation and texture coordinate optimization. The first step generates a global texture map by UV parameterization of the 3D mesh and blending of key frame images. The second step refines the generated global texture map by optimizing an objective function that imposes photometric consistency on blended images.

4.5.1 Global texture map generation

We generate a single global texture map for the simplified 3D mesh $\mathbf{M}$ by blending the key frames $\{I_i\}$. Let $\{v_i\}$ and $\{f_i\}$ denote the vertices and faces of $\mathbf{M}$, respectively. First, for each face f, we find a set of key frames in which f is visible. We call this set *sub-textures* of f. At the same time, as Fig. 4.5 shows, we estimate the 2D sub-texture coordinates of vertices of each face f by back-projecting the vertices onto each sub-texture of f. As each face has multiple sub-textures, a vertex v_i of face f has multiple sub-texture coordinates u_{ij} that are the projected coordinates of vertex v_i onto sub-textures I_j. Then each sub-texture can be mapped onto the mesh $\mathbf{M}$ using the sub-texture coordinates.

After we have estimated the sub-texture coordinates of all vertices, we generate a global texture map by blending the sub-textures of each face. Blending weight of a sub-texture I_j for vertex v_i is defined as $w_{ij}^b = \mu \cos(\theta)$, where θ is the angle between the surface normal of v_i and the view direction of the camera

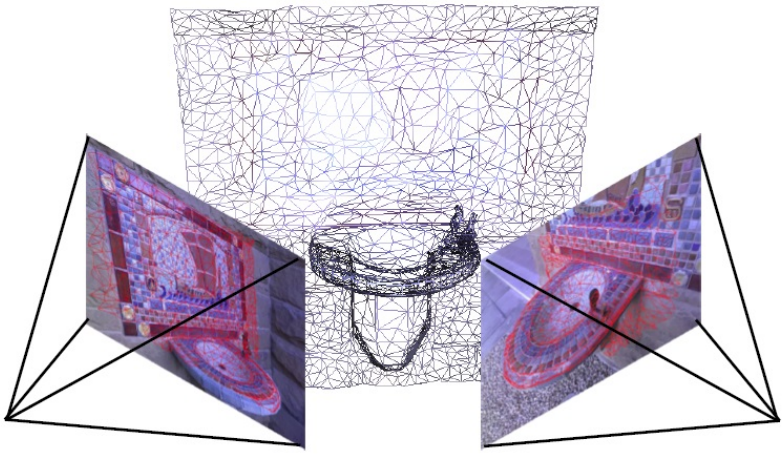


Figure 4.5: Sub-texture coordinate estimation. Each vertex is projected onto the sub-textures (key frames) to estimate the corresponding 2D sub-texture coordinates.

for I_j, and μ is a smooth weighting function that gives a higher weight when the sub-texture coordinates u_{ij} is close to the image center of I_j.

After the blending, the blended sub-texture of each face is copied onto the global texture map, which is a huge image that contains all blended sub-textures. As the reconstructed 3D mesh does not have its own texture mapping information, we should generate texture coordinates of all vertices using UV parameterization. In our method, texture coordinates are determined using the least squares conformal map generation method [64] that minimizes angle deformations of surface triangles during the parameterization. Fig. 4.6 shows the generated global texture map and sub-textures that have been blended for the *fountain* dataset.

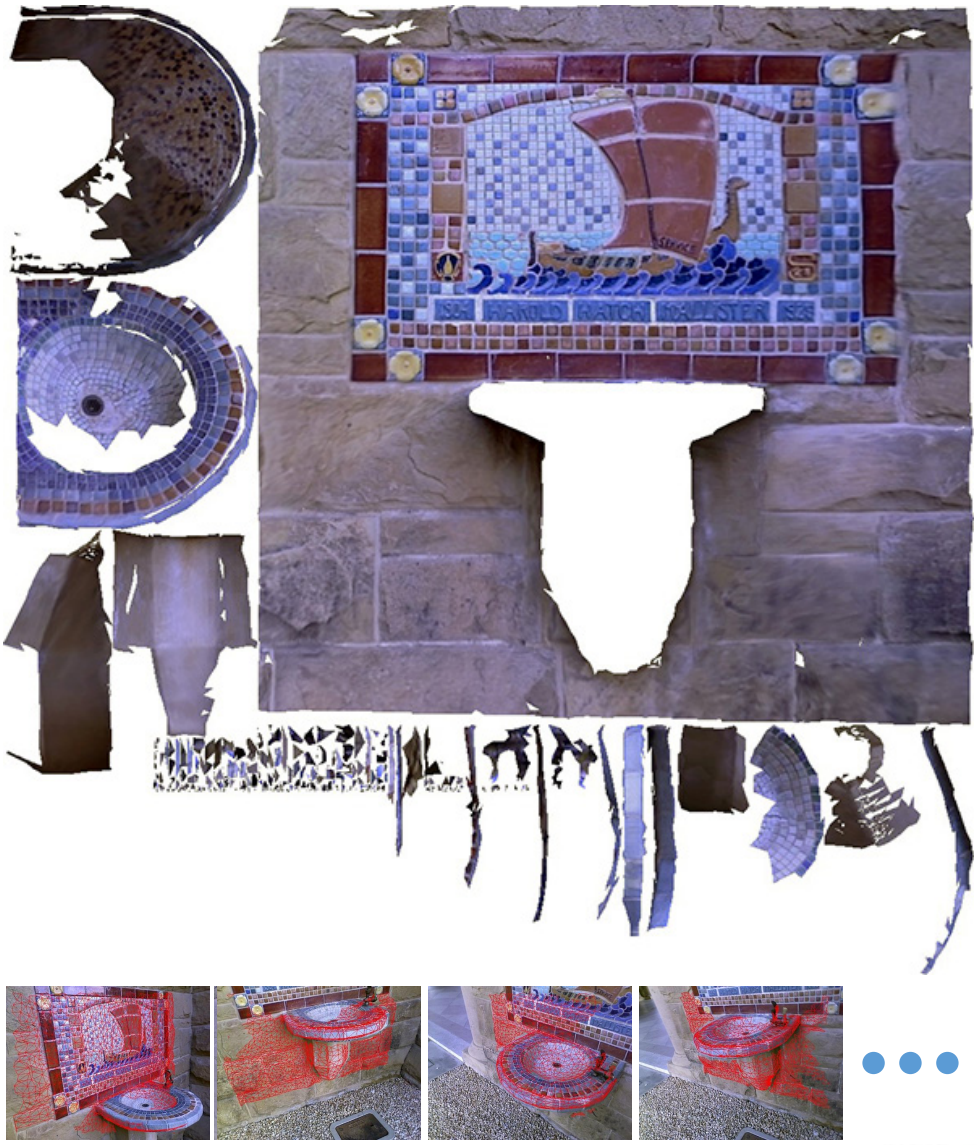


Figure 4.6: Generated global texture of the *fountain* dataset and some of sub-textures blended for the global texture.

Although we sampled the key frames considering spatial redundancy, there still can be too many key frames from which a face is visible. For computational efficiency, we limit the number of sub-textures for each face. We choose the top k key frames that have the highest blending weights w_{ij}^b for the sub-textures. We used $k = 10$.

The initially generated global texture map may suffer from blur and ghosting artifacts, which are caused by an imprecisely reconstructed 3D model and optical distortions in color images. To eliminate these artifacts and enhance the rendering quality, we optimize the global texture map by updating the sub-texture coordinates of vertices to maximize the photometric consistency among the overlapping sub-textures.

4.5.2 Global texture map optimization

Differently from Zhou and Koltun's method [21] that optimizes the non-rigid transforms of key frames, we directly optimize the sub-texture coordinates of each vertex of the simplified mesh $\mathbf{M}$. Since $\mathbf{M}$ contains relatively larger faces than the initial dense mesh $\mathbf{M^0}$, measuring the photometric consistency at only vertices would not suffice to refine the global texture map. To consider the consistency over the entire mesh, we choose a few sample points on each face. Let $S_i(f, w)$ denote sub-texture coordinates on I_i of a sample point from face f, which are determined by linear combination of the sub-texture coordinates of the three vertices of f with a combination weight w.

Our objective is to minimize the inconsistency among the sub-textures by modifying the sub-texture coordinates of vertices. Let $\mathbf{u}$ denote the concatenation of all sub-texture coordinates that we want to optimize. Also, let $\mathbf{F} = \{f_i\}$ and $\mathbf{V} = \{v_i\}$. We can quantify the sum of photometric inconsistency of all sample

points on $\mathbf{F}$ as:

$$E(\mathbf{u}) = \sum_{f \in \mathbf{F}} C(f), \tag{4.3}$$

where $C(f)$ is inconsistency among sample points on the sub-textures $\mathbf{I}_f$ belonging to a face f. We compute $C(f)$ as the inconsistency sum of image pairs in $\mathbf{I}_f$:

$$C(f) = \sum_{I_i, I_j \in \mathbf{I}_f} M(f, I_i, I_j), \tag{4.4}$$

where $M(f, I_i, I_j)$ is a function that measures the inconsistency of sample points on f between sub-textures I_i and I_j. It is defined as:

$$M(f, I_i, I_j) = \sum_{w \in \mathbf{w}} \big(\Gamma_i(S_i(f, w)) - \Gamma_j(S_j(f, w))\big)^2, \tag{4.5}$$

where $\Gamma_i(u)$ is the color value at the 2D texture coordinates u on I_i, and $\mathbf{w}$ is the set of combination weights for the sampled points on f. As $S(f, w)$ is a function of sub-texture coordinates of vertices, we can obtain optimal sub-texture coordinates $\mathbf{u}$ by minimizing Eq. (4.3).

To efficiently minimize the objective function in Eq. (4.3), we introduce two modifications. First, we re-arrange the objective with respect to the vertices rather than faces. Let $\mathbf{F}_v$ be the 1-ring neighbor faces of a vertex v, which is the set of faces that contain v as one of their three vertices. Then the modified objective can be written as:

$$E(\mathbf{u}) = \frac{1}{3} \sum_{v \in \mathbf{V}} \sum_{f \in \mathbf{F}_v} C(f). \tag{4.6}$$

Each face consists of three vertices, so Eqs. (4.3) and (4.6) are equivalent.

Second, we introduce an auxiliary variable $P(f, w)$, which is the proxy color of a sample point on f with a combination weight w. Then we re-formulate Eqs. (4.4) and (4.5) to:

$$C(f) = \sum_{I_i \in \mathbf{I}_f} M(f, I_i), \tag{4.7}$$

$$M(f, I_i) = \sum_{w \in \mathbf{w}} \big(\Gamma_i(S_i(f, w)) - P(f, w)\big)^2. \tag{4.8}$$

As a result, our objective can be written as $E(\mathbf{u}, \mathbf{P})$, where $\mathbf{P}$ is the concatenated vector of all auxiliary variables $P(f, w)$. For simplicity, in the optimization process, we use grayscale values of the key frame images, so $\Gamma_i(S_i(f, w))$ and $P(f, w)$ are scalar values.

4.5.3 GPU-based alternating solver

Our objective $E(\mathbf{u}, \mathbf{P})$ is a non-linear least squares function of sub-texture coordinates $\mathbf{u}$ and auxiliary variables $\mathbf{P}$, which can be minimized using the Gauss-Newton method. Similar to Zhou and Koltun [21], we can also use alternating optimization that optimize $\mathbf{u}$ and $\mathbf{P}$ separately while fixing the other. However, for a large reconstructed model such as a whole room, we need more drastic acceleration because the number of parameters of the objective, even for the simplified mesh, can be huge, up to $k|\mathbf{V}|$. To optimize the objective with such a large number of parameters in practical computation time, we optimize the objective function in parallel by exploiting the locality of the problem.

Our alternating optimization consists of two stages. At the first stage, we optimize $\mathbf{P}$ while $\mathbf{u}$ is fixed. Then it can be easily shown that the objective is minimized when

$$P(f, w) = \frac{1}{|\mathbf{I}_f|} \sum_{I_i \in \mathbf{I}_f} \Gamma_i(S_i(f, w)), \tag{4.9}$$

for all $f \in \mathbf{F}$ and $w \in \mathbf{w}$. Values of $P(f, w)$ are independent of each other, so they can be computed quickly in parallel.

At the second stage, we optimize $\mathbf{u}$ while $\mathbf{P}$ is fixed. This reduces to a non-linear least square problem as follow:

$$E(\mathbf{u}) = \frac{1}{3} \sum_{v \in \mathbf{V}} \sum_{f \in \mathbf{F}_v} \sum_{I_i \in \mathbf{I}_f} \sum_{w \in \mathbf{w}} r(f, I_i, w)^2, \tag{4.10}$$

where $r(f, I_i, w)$ is the residual considered in Eq. (4.8):

$$r(f, I_i, w) = \Gamma_i(S_i(f, w)) - P(f, w). \tag{4.11}$$

Then single Gauss-Newton update is defined as:

$$\mathbf{u}^{a+1} = \mathbf{u}^a + \Delta\mathbf{u}, \tag{4.12}$$

where $\Delta\mathbf{u}$ is computed as the solution of the equation:

$$J(\mathbf{u}^a)^T J(\mathbf{u}^a)\Delta\mathbf{u} = -J(\mathbf{u}^a)^T R(\mathbf{u}^a). \tag{4.13}$$

R is the residual vector that is the concatenation of $r(f, I_i, w)$ for all f, I_i, and w. J is the Jacobian of R.

Although the Jacobian matrix J is quite sparse, its size is vast for a large 3D model. For example, the reconstructed 3D model of the *lounge* dataset in Section 3.6 consists of more than 2M (millions) vertices originally. Even if we apply mesh simplification to the model, the number of vertices should be more than 0.1M for preserving the details. Then the number of parameters in the optimization goes up to 1M and the dimension of the Jacobian matrix becomes 1M×1M. Solving such a large linear system on GPU is non-trivial, and the optimization is not readily parallelizable.

To parallelize the optimization, we exploit the locality of the problem. As in Section 4.5.2, let $\mathbf{F}_v$ be the 1-ring neighborhood of a vertex v. A change of the sub-texture coordinates of v only affects the consistencies, i.e., residuals, of sample points on the faces in $\mathbf{F}_v$. In other words, the consistency among sub-textures around v is only determined by $\mathbf{F}_v$. Thus, we can subdivide the entire Gauss-Newton update problem into smaller independent sub-problems that only update the sub-texture coordinates of single vertices. These small sub-problems can be solved by performing a parallel variant of *Schwarz Alternating Method*

[65] that updates the inner variables (texture coordinates of v) using a Gauss-Newton update while keeping the boundary variables (texture coordinates of 1-ring neighbor vertices of v).

Specifically, the parallel Gauss-Newton update works as follows. As described above, each Gauss-Newton update needs to solve the linear system which is a function of the residuals. If we ignore all the residuals that are independent of a vertex v and assume the sub-texture coordinates of other vertices than v in $\mathbf{F}_v$ are fixed, then the linear system in Eq. (4.13) becomes very small, and evaluating the residual and numerical calculation of the inverse of a 2×2 matrix is only required. With this simplification, we can use single kernel threads on GPU to perform parallel Gauss-Newton updates for the sub-texture coordinates of vertices. To propagate the updated sub-texture coordinates to the 1-ring neighborhood aggressively, we update $\mathbf{u}$ twice for each $\mathbf{P}$ update. In Section 3.6, experimental results show the effectiveness of our GPU-based alternating solver, which can handle optimization for a large 3D model within tens of seconds.

4.6 Experimental Results

We performed various experiments to evaluate the proposed method. We tested our method on several 3D reconstructed models with RGB-D streams (Fig. 4.7), provided by Zhou and Koltun [5, 21], which were taken using an Asus Xtion Pro Live RGB-D camera. Except the *fountain* data, we used 3D meshes and camera trajectories that were estimated by [5]. For the *fountain* data, we used voxel hash based reconstruction [6] to reconstruct the 3D model and estimated the camera trajectories from the given RGB-D images. All experiments were performed on a PC with an Intel i7 4.0GHz CPU, 16GB RAM and Nvidia GeForce GTX TITAN X GPU. The implementations of algorithms [62] and [64] in MeshLab [66] and Blender [67] were used for mesh simplification and UV

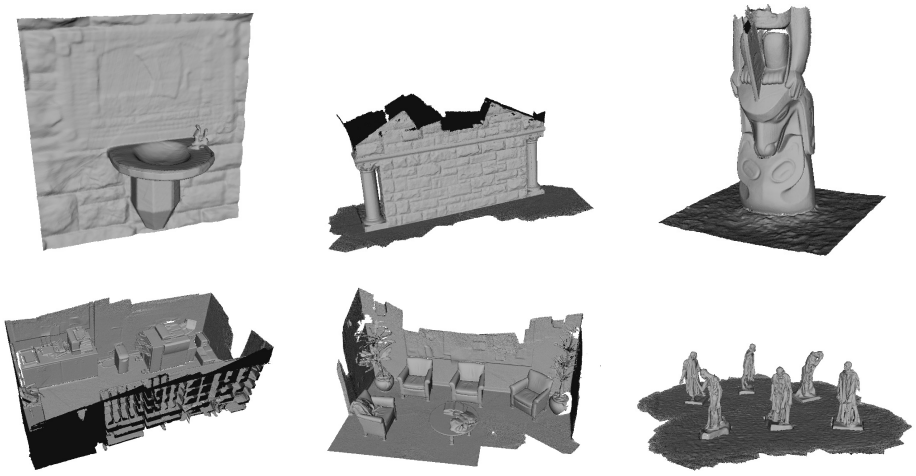


Figure 4.7: 3D reconstructed models used in our experiments: from top left, *fountain, stonewall, totempole, copyroom, lounge, burghers*. The models were reconstructed using the methods of Zhou and Koltun [5, 21].

parameterization of a simplified mesh, respectively.

Timing data Our method refines the global texture map using iterative optimization. Fig. 4.8 shows that the rendering results are progressively refined through the optimization of the sub-texture coordinates. The blurry texture map becomes sharp in only few tens of iterations. We used 100 iterations for all experiments in this chapter. Table 4.1 shows computation time and other statistics of our method. Our spatio-temporal key frame sampling and the initial global texture generation takes a few seconds, depending on the size of the image stream. Texture map optimization takes several to tens of seconds, which is much faster than Zhou and Koltun [21]. For example, for the *fountain* model, Zhou and Koltun [21] took more than 200s according to their paper, but our optimization only takes 2.6s, taking 26ms per iteration.

Scalability With the GPU-based alternating solver, our approach has much higher scalability than [21]. Zhou and Koltun [21] formulated the optimization problem as n independent linear systems with 720 variables, where n is the num-

Model	# of images	# of key frames	# of original faces	# of simplified faces	Time for optimization
fountain	1,086	27	532,806	10,000	2.6s
totempole	2,700	103	1,285,993	10,000	2.5s
stonewall	2,700	113	4,345,379	65,000	9.5s
lounge	3,000	106	3,150,436	130,000	16s
copyroom	5,490	155	5,062,748	130,000	18s
burghers	11,230	319	6,858,620	195,000	31s

Table 4.1: Test dataset statistics and the computation time of our method. For a large-scale model, the proposed GPU-based alternating optimization takes only few tens of seconds to obtain a photometrically enhanced global texture map.

ber of key frames. In that case, different linear systems cannot be solved in parallel on a GPU because solving each system would need multiple kernel threads. In contrast, exploiting the locality, our GPU-based alternating solver separates the problem into m independent linear systems with only two variables, where m is the number of vertices of the simplified mesh. Many of these small linear systems can be solved in parallel on a GPU, as each system can be handled by a single kernel thread with few arithmetic operations.

Mesh simplification We first evaluate the robustness of our method against mesh simplification in terms of the rendering quality. As described in Section 4.5.3, the processing time of texture coordinate optimization depends on the number of vertices in the simplified mesh, so we can achieve more acceleration with aggressive mesh simplification. Fig. 4.9 demonstrates that the rendering quality is still satisfactory even when the number of faces is extremely reduced to 1% of the original.

Adaptive sampling We also evaluate the effectiveness of our spatio-temporal adaptive sampling. Temporal sampling based on the blurriness reduces the num-

initial

5 iterations

50 iterations

Figure 4.8: Progress of the iterative texture optimization on a cropped region of the *fountain* model. As the objective function is minimized, the photometric consistency of the generated texture map is improved drastically. Best viewed on a color monitor.

ber of key frames depending on the length of the input stream. Then the spatial sampling based on uniqueness reduces the key frame set based on the scale of the scene. It shortens the processing time of texture map optimization by eliminating redundant images which are not necessarily useful for reconstructing the 3D model. Fig. 4.10 shows the rendering results when only temporal sampling is applied and when both temporal and spatial samplings are applied. By perform-

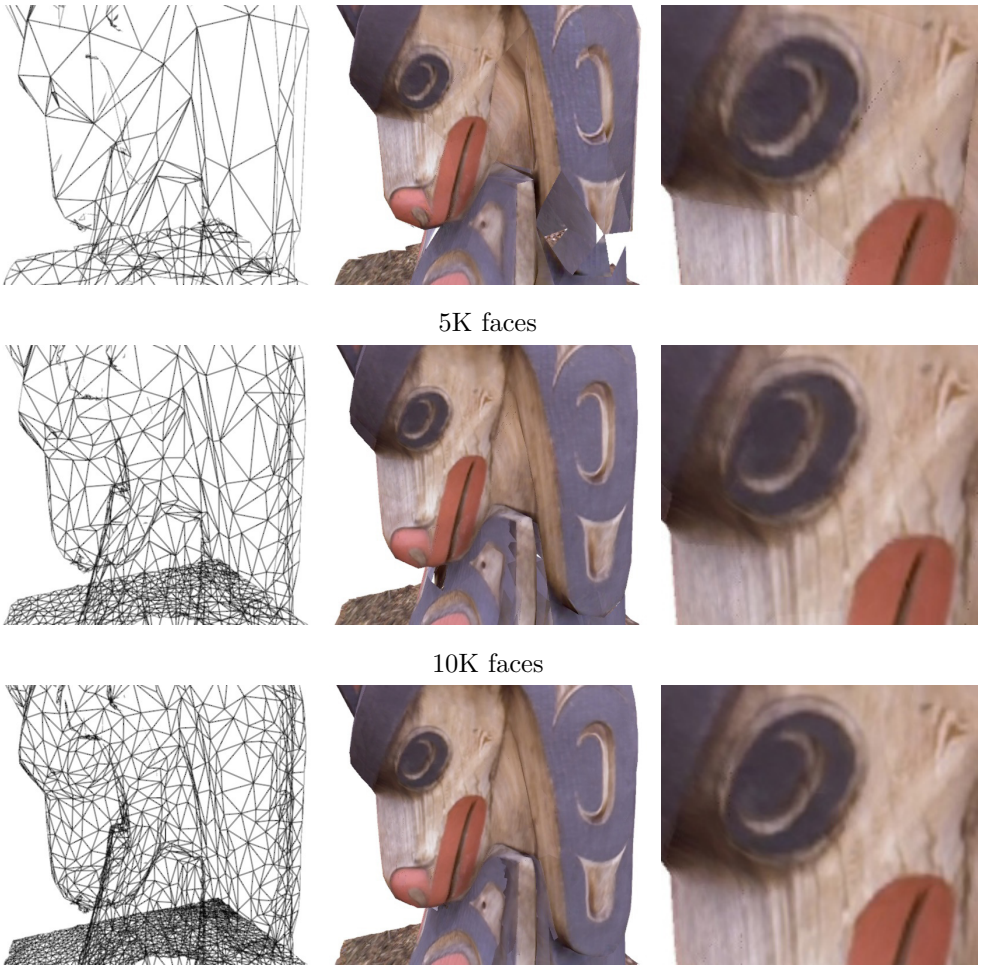


Figure 4.9: Rendering results of the *totempole* model simplified to different numbers of faces. Even with only 10K faces, the rendering quality is not much degraded while the optimization takes less than 3 seconds. Note that the original mesh consists of more than 1M faces.

ing spatial sampling, we reduced the number of sampled key frames from 214 to 113 while preserving the rendering quality. Note that the images participating in the sub-texture blending could have been changed due to the spatial sampling, causing some color differences between the rendering results.

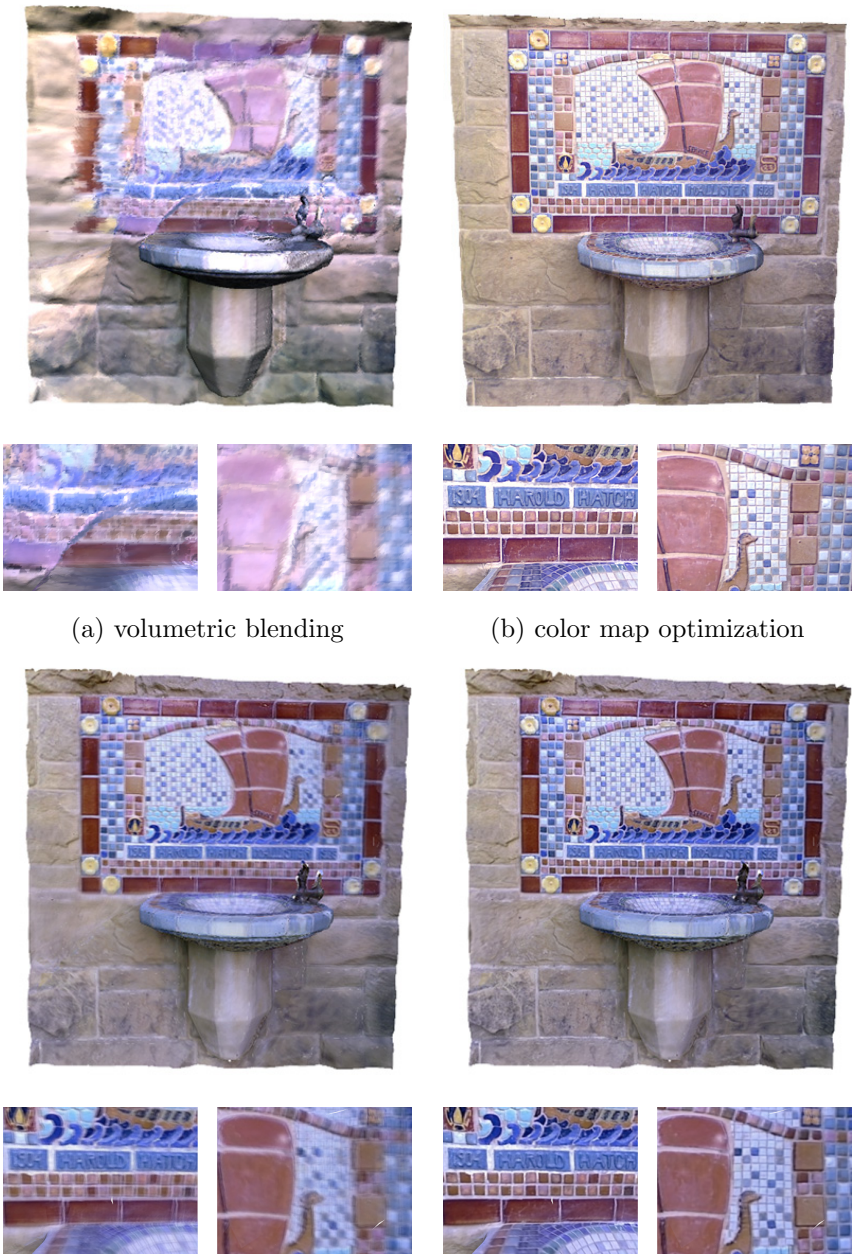
Visual comparison For the *fountain* model, we compared our result with Zhou and Koltun [21] using the reconstructed mesh of [21] provided by the authors.

w/o spatial sampling w/ spatial sampling

Figure 4.10: Rendering results with/without spatial sampling. Spatial sampling removes a half of key frames, which are redundant, preserving the rendering quality.

Fig. 4.11 shows visual comparison. The result in [21] was generated from a high-resolution (1920 ×1080) color image stream, which was not available to us. Instead, we used a low-resolution (640 × 480) stream provided by the authors to generate our global texture map. Nevertheless the rendering results are quite comparable, while our processing time is about 100× faster. Note that the color balance of the rendering result in [21] differs from ours because the input color streams are not identical. We applied auto white-balancing to our input color stream to compensate the difference.

Large-scale model Due to the adaptive key frame sampling and GPU-based alternating optimization, our approach is scalable to handle large-scale 3D reconstruction. We demonstrate the scalability of our method by generating texture maps for reconstructed 3D indoor scenes. We tested three models, *copyroom, lounge, burghers*, which consist of more than 5M faces on average. Each model is simplified to 130K or 195K faces based on the geometric complexity. Fig. 4.12-4.14 shows several rendering results from different viewpoints. These examples demonstrate that our method can be used to generate precise texture maps needed for visualizing large-scale 3D reconstructed indoor scenes.



(a) volumetric blending (b) color map optimization

(c) initial texture map (d) optimized texture map

Figure 4.11: Rendering result of the *fountain* model with our generated texture map. We compare the result with the volumetric blending approach [6] and the color map optimization [21]. Note that the overall difference of color balances between [21] and ours has come from the difference in the input image streams.

Figure 4.12: Texture map generation and optimization results for a large-scale 3D reconstructed model (*lounge*). White holes in the rendering results have come from the missing geometry of the reconstructed models.

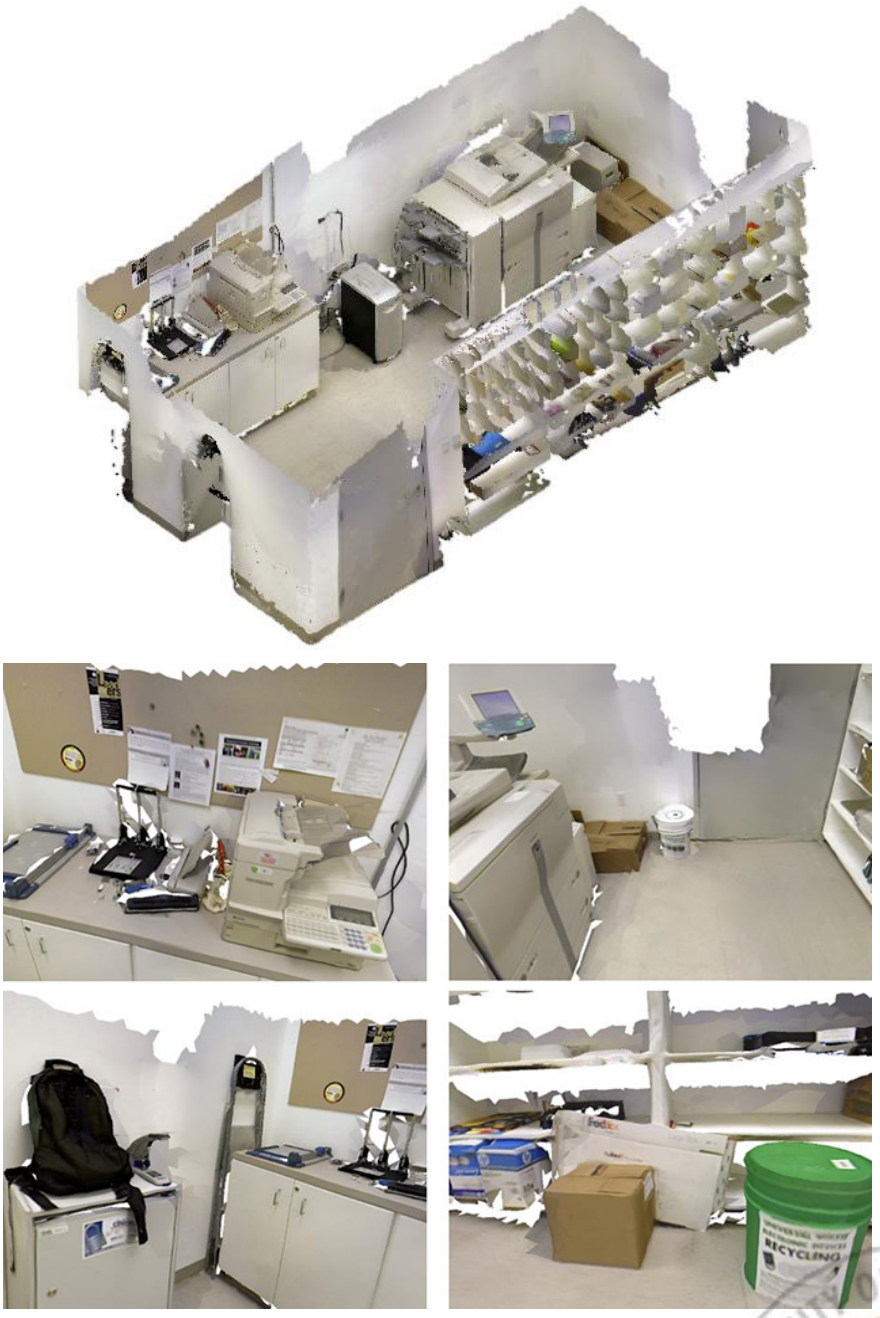


Figure 4.13: Texture map generation and optimization results for a large-scale 3D reconstructed model (*copyroom*). White holes in the rendering results have come from the missing geometry of the reconstructed models.

Figure 4.14: Texture map generation and optimization results for a large-scale 3D reconstructed model (*burghers*). White holes in the rendering results have come from the missing geometry of the reconstructed models.

4.7 Discussion

Differently from previous works based on voxel or vertex colors, we proposed a texture-mapping based approach for representing and rendering color information of a 3D reconstructed model. Reconstructed geometric models, especially indoor scenes, equipped with precisely generated texture maps can be used as 3D model contents for various applications, such as virtual reality and 3D fabrication.

Limitation and future work Our algorithm generates a single global texture map for a whole 3D reconstructed scene. When the algorithm is applied to a large-scale scene, the required size of the global texture map could become too large. Subdividing a large model into small ones and generating separate texture maps can resolve this issue. Our texture map optimization method only considers local information (1-ring neighborhood) of each vertex to enhance the photometric consistency. Exploiting this locality enables our method to work very fast with parallel implementation, but prohibits the method from working with a dense mesh where local information is limited. As shown in Fig. 4.6, our global texture map currently contains lots of holes on which no vertices are assigned. Advanced mesh parameterization methods [68, 69] would be useful to resolve this problem.

V. Semantic Reconstruction: Reconstruction of Semantically Segmented 3D Meshes via Volumetric Semantic Fusion

5.1 Motivation

Semantic segmentation is one of the challenging problems for high-level scene understanding, which has various applications such as autonomous robots and augmented reality. In recent years, semantic segmentation has been spotlighted and its performance has been drastically improved along with rapid advances of deep learning. While plausible semantic segmentation results could be obtained for 2D images of indoor as well as outdoor scenes, semantic segmentation of 3D scene models still remains a hard problem.

The success of 2D semantic segmentation is largely built upon the availability of huge labeled image datasets and advances in knowledge transfer techniques. For example, Microsoft COCO dataset [70] contains about 50K semantically annotated 2D images. Although the dataset size may not be enough to train a deep convolutional neural network with millions of parameters from scratch, knowledge transfer learning makes it possible to exploit the features learned from a huge number of images, such as ImageNet dataset [71], for finetuning a network for semantic segmentation.

On the other hand, in the case of 3D semantic segmentation, it becomes more challenging to develop a deep learning based approach. Although several semantically annotated 3D scanned scene datasets have been recently released [72, 73, 74], their sizes are about hundreds to a thousand, which are not large enough to train complex neural networks. In addition, transfer learning is not easily

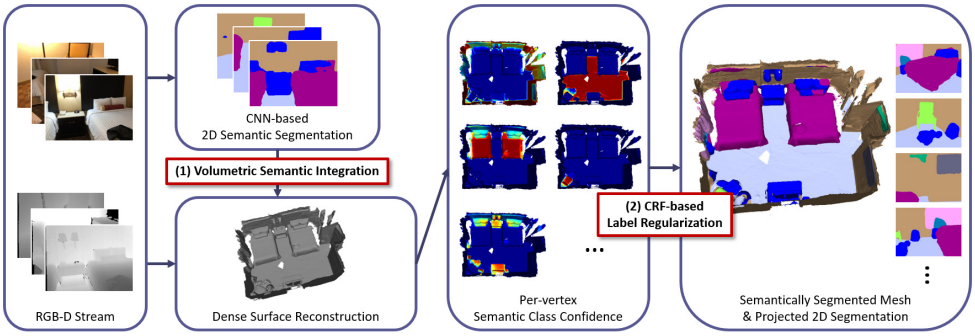


Figure 5.1: Our pipeline for semantic segmentation of reconstructed 3D indoor scenes. 2D semantic segmentation results, as well as the geometry information, of the input RGB-D stream are fused into a volumetric representation, and then per-vertex semantic class confidences are extracted when the mesh is reconstructed from the volumetric representation. The per-vertex confidences are refined through CRF-based label regularization, generating the final semantic segmentation of the reconstructed mesh. Projections of the semantically labelled reconstructed mesh can be used for producing semantic segmentations of input frames, which are more accurate than the initial segmentations of the frames fused into the volumetric representation.

applicable to finetune a 3D CNN for semantic segmentation as there is no such a huge general 3D dataset similar to ImageNet [71].

Despite these difficulties, a few approaches based on deep learning have been proposed for 3D semantic segmentation, which can be categorized into two groups: direct and indirect methods. To obtain a semantic map of the entire 3D scene, a direct method performs semantic label estimation directly on the 3D model, usually represented in a voxelized format [72] or point cloud [75, 76, 77]. In contrast, an indirect method integrates 2D semantic segmentation predictions onto the 3D model, represented in the form of surfels [78] or a point cloud [79, 80]. Direct methods require a large-scale 3D labeled training dataset, which would be harder to be annotated than 2D data. In addition, semantic segmentation results of an indirect method based on surfels or a point cloud would need further processing to be used for reconstructed 3D meshes.

In this paper, we propose a novel framework that automatically generates semantically segmented triangle meshes from a RGB-D stream for a large-scale indoor scene by combining 2D semantic segmentation with 3D volumetric fusion. Differently from prior methods [72, 79, 80, 78] that use voxels, surfels, or a point cloud, our method utilizes a volumetric data structure and a triangle mesh complementarily to integrate geometry and semantic information. The volumetric data structure enables us to efficiently integrate the high-level 2D semantic information from a state-of-the-art deep learning technique onto the reconstructed geometry, and the connectivity of the triangle mesh is utilized for CRF-based semantic label regularization that enhances the segmentation result by restoring miss-labeled parts and removing noisy labels.

The key characteristics of our framework include:

- range-sensitive volumetric semantic fusion method that incrementally integrates CNN-based 2D semantic segmentation results onto the densely reconstructed 3D geometry.
- CRF-based semantic label regularization using the geometric and photometric information of the reconstructed triangle mesh to incorporate the global scene context.

Experimental results on various RGB-D streams of large-scale indoor scenes show that our method can precisely predict the dense (i.e., per-vertex) semantic labels for a reconstructed mesh without directly training a deep neural network on a 3D dataset. As applications of the segmentation results, we present 3D scene completion and manipulation, where semantic information is used for detecting and filling holes inside objects and transforming objects in the scene independently from others.

5.2 Related Work

RGB and RGB-D image semantic segmentation Deep convolutional neural networks (CNNs) have been successfully used for semantic segmentation of single RGB images [81, 82, 83, 84, 85, 86]. For 2.5D RGB-D images, Long et al. [83] trained and tested their fully convolutional networks (FCNs) on the NYU-Depth V2 dataset [87]. Hazirbas et al. proposed FuseNet [88] that integrates the intermediate depth and color features using sparse and dense fusion. Park et al. proposed RDFNet [89] that uses multi-modal feature fusion to incorporate the depth information into semantic segmentation and shows the state-of-the-art performance.

To integrate global context to the final semantic segmentation prediction, Conditional Random Fields (CRFs) can be adopted as a post-processing step. Krähenbühl and Koltun [90] proposed an efficient approximate algorithm for fully connected pairwise CRF to refine semantic segmentation results. It uses pairwise Gaussian edge potentials considering the distances based on pixel positions and appearances.

Full 3D semantic segmentation Several works have been proposed for semantic segmentation of scanned full 3D geometric data. Huang et al. [91] firstly proposed a method that uses a 3D CNN by voxelizing a point cloud into a regular grid. Dai et al. [72] released 15K semantically annotated triangle meshes, and trained a 3D CNN by voxelizing the meshes. Instead of a voxelized regular grid, recently proposed point cloud based methods [75, 76, 77] directly estimate the class labels of an unordered set of points. RSNet [77] achieved the state-of-the-art results by modeling local geometric dependencies.

To avoid direct training of 3D CNNs, indirect segmentation methods have been proposed. Lawin et al. [80] projects a 3D point cloud onto a set of synthetic

2D images, which are used to predict semantic labels of the projected points using a 2D CNN. McCormac et al. [78] proposed a 3D semantic mapping method that predicts semantic maps from RGB-D frames and fusing them onto the surfel data structure.

Although indirect 3D semantic segmentation methods [80, 78] have similar pipelines that fuse 2D predictions onto a single geometry, our method differs from them in that we use a volumetric representation for intermediate fusion and produce a triangle mesh with per-vertex class labels as the output.

Dense 3D surface reconstruction After KinectFusion [14] was introduced, volumetric integration [92] of a signed truncated distance function (TSDF) becomes a common approach for reconstructing the geometry for RGB-D video based 3D scanning [19, 17, 20]. VoxelHashing [17] uses a hash-based volumetric structure to enable the reconstruction of a large-scale indoor scene. BundleFusion [20] shows the state-of-the-art performance in real-time 3D reconstruction, which can handle incremental drifts in pose estimation using color features. Our framework uses BundleFusion [20] and VoxelHashing [17] for pose estimation and geometry reconstruction, respectively.

RGB-D image dataset As consumer depth cameras become popular, several RGB-D image datasets have been published for indoor scene reconstruction and understanding. NYUDv2 dataset [87] consists of semantically annotated thousands of RGB-D images, and SUN RGB-D dataset contains 10K images. ScanNet dataset [72] is now the largest RGB-D image dataset and consists of 2.5 million images. Especially, ScanNet contains a semantically labeled triangle mesh for each input RGB-D stream. In this paper, we use ScanNet dataset for experiments.

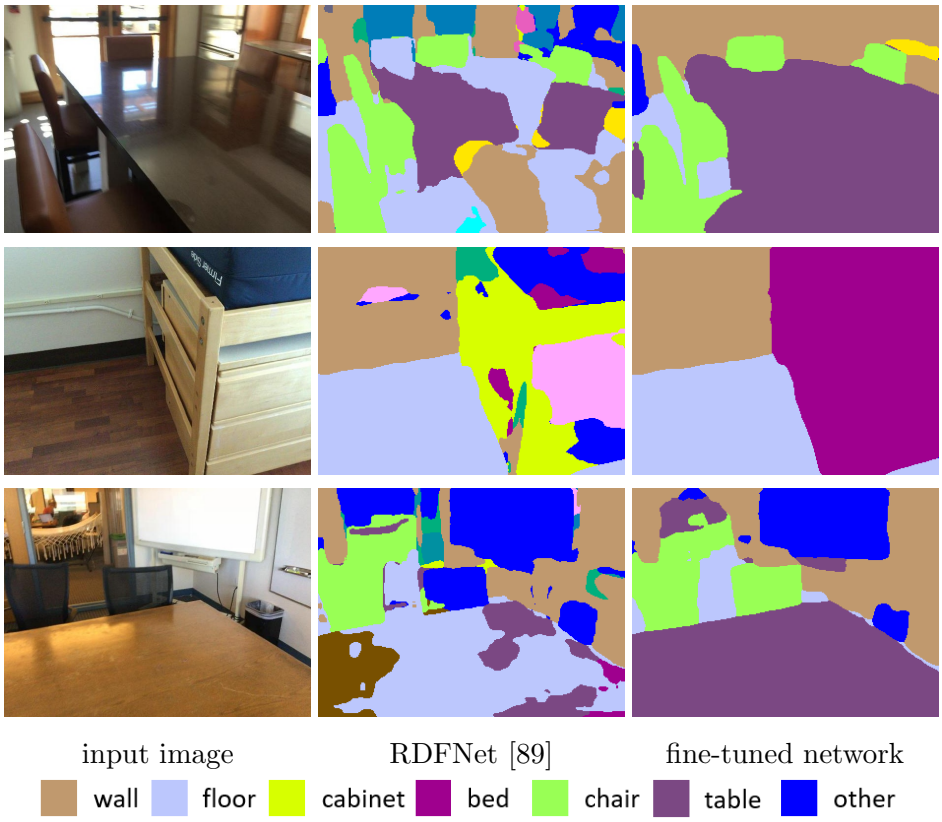


Figure 5.2: Results of 2D CNN-based semantic segmentation. The network fine-tuned on ScanNet dataset [72] shows cleaner and more accurate segmentation results compared to the original RDFNet [89].

5.3 Our Framework

The overall process of our approach for 3D semantic segmentation is shown in Fig. 1. Our framework uses a RGB-D image stream as the input. During the semantic class confidence integration step, we apply CNN-based 2D semantic segmentation to the input RGB-D steam to obtain a class confidence map for each frame. Then the estimated class confidence maps are fused into a 3D volumetric data structure along with the geometry during the reconstruction process. In the semantic mesh generation step, we extract a 3D triangle mesh from the integrated volume data structure. At the same time, we extract per-vertex class confidences for object classes using a modified marching cube algorithm [93].

The per-vertex object class labels are finally determined by fully-connected CRF inference using the unary and pairwise potentials considering local and global context information.

Dense volumetric reconstruction In our work, we use BundleFusion [20] to estimate the camera poses of the input RGB-D stream. BundleFusion [20] robustly tracks the camera poses of incoming RGB-D frames and minimizes the accumulated drift of the geometry by globally optimizing the tracked camera poses based on geometric and photometric feature matching. We then perform the volumetric integration of geometry and semantic information using VoxelHasing [17]. It employs a sparse volumetric grid based on spatial hashing as the geometry representation which enables us to handle a large-scale indoor scene.

5.4 CNN-based 2D Semantic Segmentation

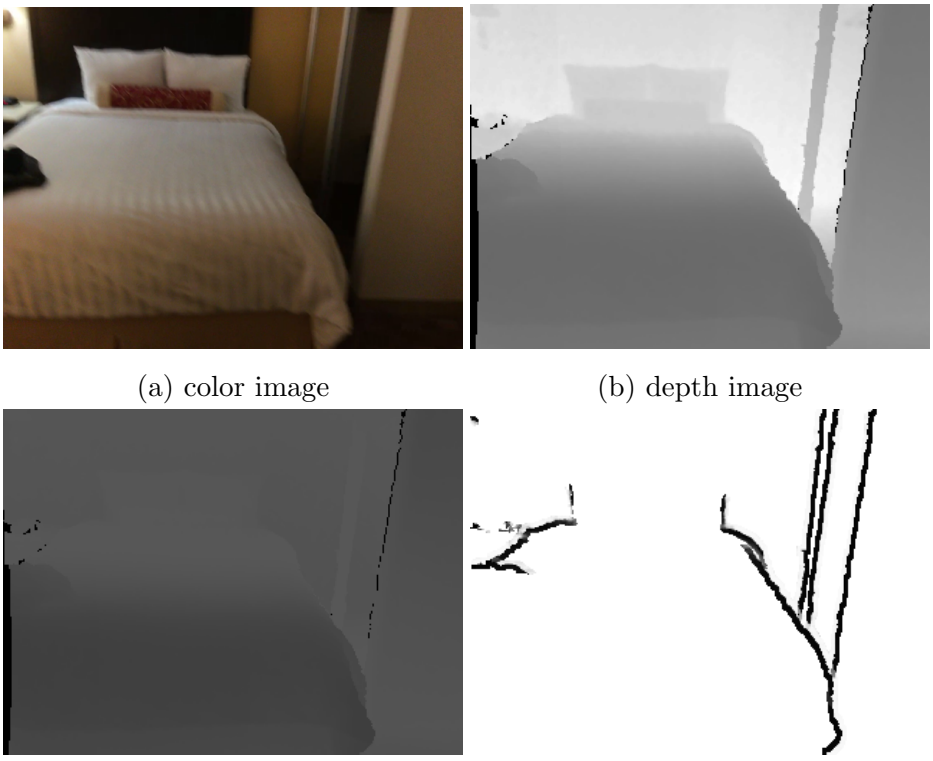
In this section, we present the details of our single image semantic segmentation step. Given a RGB-D stream, we apply a CNN-based semantic segmentation method to each frame. The network is fine-tuned on the annotated RGB-D images from ScanNet dataset [72] to improve the segmentation quality.

Semantic segmentation network Semantic segmentation of input frames is an independent component in our framework, and we could use any 2D image segmentation methods [81, 83] for the step. However, we found that depth features help better discriminate ambiguous semantic classes (e.g. floor and table top surface), and a RGB-D semantic segmentation method would work better. In this paper, we use RDFNet [89] that effectively exploits multi-level RGB-D CNN features and learns the optimal fusion of multi-modal features. RDFNet shows the state-of-the-art accuracy for RGB-D semantic segmentation of indoor scenes tested on NYUDv2 dataset [87].

Network transfer learning Although RDFNet [89] reports the state-of-the-art performance on RGB-D semantic segmentation, it does not always produce satisfactory results depending on the property of the input scene. RDFNet is trained on NYUDv2 dataset [87], many of whose images were captured moderately far from the target objects covering the whole scenes. On the other hand, ScanNet dataset [72] is constructed for dense surface reconstruction, and the images in the dataset were usually captured close to the target objects, containing only parts of large objects, such as beds or tables (Fig. 5.2). Furthermore, the RGB images may suffer from motion blurs as they were captured using hand-held RGB-D sensors. Consequently, the original RDFNet shows inferior performance on ScanNet dataset.

To compensate this limitation, we perform transfer learning for RDFNet on ScanNet dataset. We extract 16K RGB-D images and their corresponding semantic annotations from 1041 training streams by sampling every 100th frame. Based on the author-provided pre-trained model of RDFNet, we fine-tune the entire network for 20 epochs. We use 20 major object classes in accordance with the original paper [72] of ScanNet dataset. Following the RDFNet paper [89], random crop and horizontal flip are used for data augmentation to prevent the training from over-fitting. As shown in Fig. 5.2, the fine-tuned network shows robust semantic segmentation results, even when the input images are captured extremely close to the target objects.

Confidence vs. probability In this paper, we use the term *confidence* instead of *probability* to refer to the output values of a semantic segmentation network. It is known that modern complex CNNs may not be well-calibrated [94], which means the output values of a neural network cannot be directly interpreted as the correctness likelihoods for ground truths. Therefore, we treat the output of



(a) color image (b) depth image

(c) depth weight W^D (d) boundary weight W^B

Figure 5.3: Reliability weight maps for a RGB-D image. The depth-based accuracy weight W^D is the highest at the moderate distance, and the boundary misalignment weight W^B has low values around depth discontinuities, where a darker color means a lower weight.

a semantic segmentation network as a *confidence* of each object class.

5.5 Semantic 3D Reconstruction

In this section, we present the details of our semantic 3D reconstruction framework. Although naive implementation of the framework could be rather straightforward, it would produce noisy and less inaccurate segmentation results, not remedying inherently incomplete results of 2D semantic segmentation. To obtain high-quality semantic segmentation of the reconstructed mesh, we introduce and use the reliability of confidence values for improving semantic class confidence integration and CRF-based label regularization.

5.5.1 Semantic class confidence integration

Single-view 2D semantic segmentation predicts the segmentation result of each view independently, and it often produces noisy and unstable results, as shown in Fig. 5.2. In 3D reconstruction techniques [20, 19, 14, 17], the reconstructed geometry becomes smooth and clean as the multiple noisy depth frames are integrated incrementally. To resolve the noise and uncertainties in the single-view 2D semantic segmentation results, we use incremental semantic fusion, similarly to geometry refinement.

During surface reconstruction, each voxel holds the truncated signed distance function (TSDF) value to represent its surrounding local geometry. To integrate the semantic information of the incoming RGB-D stream, we also store the confidences of object classes at each voxel to represent which object class the voxel may belong to. The integration of class confidences follows a similar way to the original TSDF update of VoxelHashing [17].

Each pixel p of the input image has its corresponding voxel o according to the estimated camera pose and the pixel coordinates. Then given a predicted class confidence map from 2D semantic segmentation of the t-th frame, we incrementally update the class confidences of the corresponding voxels $C_t(o)$ by taking the weighted running averages [14], defined as follows:

$$C_t(o) = \frac{W_{t-1}(o)C_{t-1}(o) + W_{F_t}(p)C_{F_t}(p)}{W_{t-1}(o) + W_{F_t}(p)}, \tag{5.1}$$

$$W_t(o) = W_{t-1}(o) + W_{F_t}(p), \tag{5.2}$$

where $C_{t-1}(o)$ and $W_{t-1}(o)$ are the *integrated* class confidence and reliability weight of voxel o, respectively. $C_{F_t}(p)$ and $W_{F_t}(p)$ are respectively the class confidence and reliability of pixel p in the t-th frame. Note that a class confidence map from 2D semantic segmentation contains confidence values for all labels at

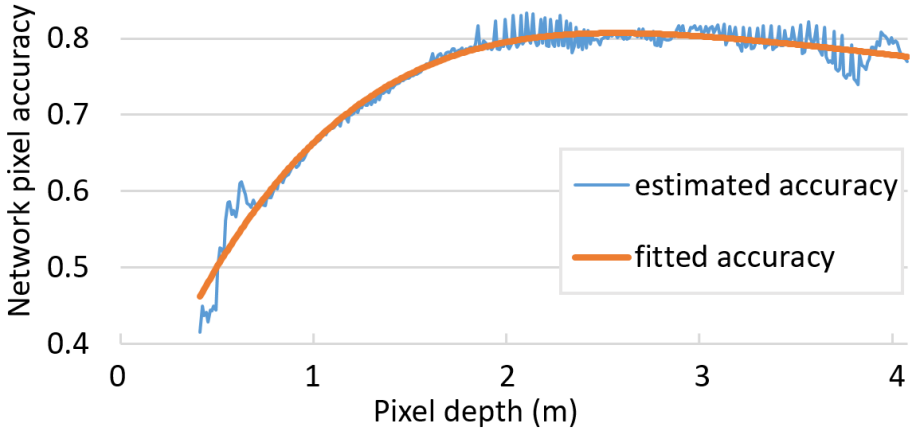


Figure 5.4: Semantic prediction accuracy along with the input pixel's depth value (blue), and its 4th degree polygonal fitting (red). Too close capturing of an object reduces the accuracy.

each pixel, whose sum is one. So $C_t(o)$ is a d-dimensional vector, where d is the number of different labels in semantic segmentation.

Although we use the state-of-the-art 2D semantic segmentation method, the semantic prediction result still contains noise and inaccurate confidence values. These artifacts vary depending on input frames, and we define the per-pixel class reliability $W_{F_t}(p)$ of the semantic segmentation result and use it for adaptively integrating the semantic predictions. $W_{F_t}(p)$ is a d-dimensional vector determined by local geometry information as:

$$W_{F_t}(p) = W^D_{F_t}(p) W^B_{F_t}(p), \tag{5.3}$$

where $W^D_{F_t}(p)$ is the depth-based accuracy weight and $W^B_{F_t}(p)$ is the boundary misalignment weight (Fig. 5.3).

Depth-based accuracy weight $W^D_{F_t}$ A CNN has a fixed receptive field size according to its network structure, and the accuracy of semantic prediction varies depending on the object scale in the input image. To reflect this, the weight $W^D_{F_t}(p)$ is defined as a function of the depth value of pixel p. Recall that the

object size in an image changes with the distance from the camera. We evaluated RDFNet [89], our choice of the 2D semantic segmentation method used in our framework, on the validation set of ScanNet dataset [72], and measured the accuracies of semantic predictions for different depth values in *1cm* intervals (Fig. 5.4). Then we fitted a 4th degree polynomial to approximate the prediction accuracy function with respect to the depth value. The function value ranges from 0 to 1, and we use it as $W^D_{F_t}(p)$.

Boundary misalignment weight $W^B_{F_t}$ Semantic segmentation results of a RGB-D stream mainly depend on the color images while depth information is supplementarily used for distinguishing ambiguous labels. However, even with a well-calibrated RGB-D sensor, there still exist misalignments between the color and depth images. Especially, such misalignments may become large along the boundaries between foreground objects and the background room layout (i.e., wall and floor) due to large depth differences. When the semantic predictions are integrated with geometry reconstruction, these misalignments would introduce mis-labeled voxels around object boundaries in the scene.

To address this issue, we introduce the boundary misalignment weight $W^B_{F_t}$, which gives a low reliability to the room layout classes (wall and floor) for the foreground pixels around depth discontinuities. Specifically, we first detect the depth edge pixels p which contain depth differences larger than *30cm* in the neighboring 7×7 windows, and then define $W^B_{F_t}(p)$ as:

$$W^B_{F_t}(p) = \frac{1}{1+\exp(-\alpha(r(p)-\beta))} \qquad r(p) = \frac{d(p)-d_{min}}{d_{max}-d_{min}} \tag{5.4}$$

where $d(p)$ is the depth of pixel p, and d_{min} and d_{max} are the minimum and maximum depths in the window centered at p, respectively. $W^B_{F_t}$ is computed with Eq. (5.4) only for the two room layout classes, i.e., wall and floor, and

remains 1 for all other classes, preventing foreground pixels from being labeled as layout classes. Note that in Eq. (5.4), $W^B_{F_t}(p)$ becomes small when the depth of pixel p is small, meaning that p tends to belong to a foreground object. For non-depth edge pixels p, $W^B_{F_t}$ is 1 for all semantic classes. In the experiments, we set $\alpha = 8$, $\beta = 0.5$.

Fig. 5.3 shows weight maps $W^D_{F_t}$ and $W^B_{F_t}$ that visualize the per-pixel reliability used for adaptively integrating a class confidence map during the reconstruction process. Moreover, at the end of reconstruction, the integrated reliability weight $W_t(o)$ of each voxel o represents the reliability of the integrated class confidence $C_t(o)$ of o, and we use it in the CRF-based label regularization.

Besides the depths of pixels and the boundary misalignments, other factors, e.g., physical object sizes, could be related to the class reliability. In our work, however, we consider the two factors only as they are most intuitive and can be easily estimated.

5.5.2 CRF-based semantic mesh generation

After integrating geometry and semantic information into a volumetric representation, we generate a semantically labelled triangle mesh from the volume using the marching cube algorithm [93]. By modifying the original marching cube algorithm, we assign the object class confidence to each vertex by linearly interpolating the confidences integrated at neighboring voxels. Fig. 5.5 shows the reconstructed mesh and assigned class confidences for five major object classes, where red and blue vertex colors represent high and low confidences of a vertex for an object class, respectively.

CRF-based label regularization Our semantic reconstruction effectively reduces the noise and uncertainty by fusing multiple semantic predictions. However, 2D semantic segmentation only considers local appearance and geometry in

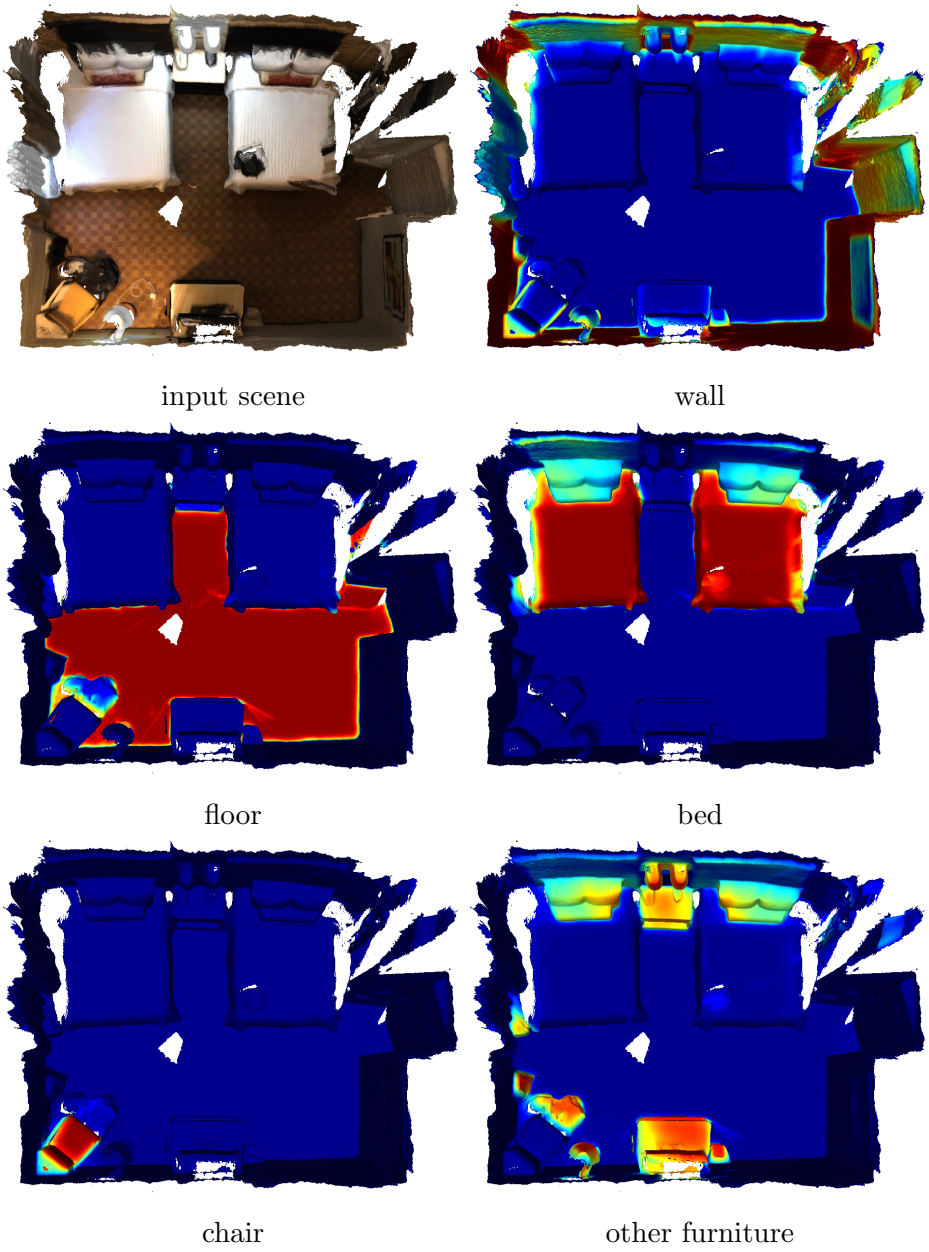


Figure 5.5: Class confidence visualization for five major classes (red: high, blue: low).

a limited field-of-view of an input frame. Simply taking the maximum value from the integrated class confidences $C(x)$ to determine the class label of a vertex x would produce a noisy segmentation result. To incorporate the global context of the reconstructed scene into semantic segmentation, we use conditional ran-

dom field (CRF), which is a common approach to refine the output of a CNN in 2D semantic segmentation. Prior 3D semantic segmentation methods [79, 78] also used CRFs on a point cloud or surfels to regularize the semantic map. In contrast, we apply CRF to the reconstructed 3D triangle mesh to determine the final semantic labels of vertices.

We construct a fully connected CRF and use the mean-field approximation algorithm [90] to efficiently solve it. In the CRF construction, we treat a 3D vertex as a graph node in the field. In a fully connected CRF, each node is connected to every other node no matter how far it is, and so we can consider the local and global scene contexts simultaneously for semantic mesh segmentation.

The labeling status of a complete CRF graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be formulated as the Gibbs energy $E(\mathbf{x})$, where $\mathbf{x} = \{x_i\}$ denotes a labeling set of all vertices $\{v_i\} \in \mathcal{V}$. The energy $E(\mathbf{x})$ is defined as combination of the unary potential ψ_u and the pairwise potential ψ_p:

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j), \tag{5.5}$$

where i and j are vertex indices.

Reliability-based unary potential ψ_u Basically the unary potential $\psi_u(x_i)$ is defined as a log-probability of a given class label for a vertex v_i which comes from the result of volumetric semantic fusion. To obtain the class probability, we use the integrated object class confidences $C(v_i)$. We also exploit the prior knowledge on the input scene in determining the class probability. We assume that there exists only one floor in the scene, and ignore the floor class confidence of the vertices above the detected floor. We first estimate the 3D plane of the floor by applying RANSAC [95] to the vertices which have high confidences on the floor class. We then reduce the confidence on the floor class to be zero for the vertices away from the floor plane.

In addition, we use the integrated reliability weight of each vertex as the certainty of the object class confidence for the unary potential. For example, the object class confidences should be adjusted to follow the uniform distribution if the reliability weight is *0*, and the confidences should keep the original distribution when the reliability is *1*. In our implementation, we use the following equation to adjust the confidence distribution:

$$C'(v) = \lambda_r W(v) C(v) + (1 - \lambda_r W(v)) p_u, \tag{5.6}$$

where $C(v)$ is the integrated confidence of vertex v and $W(v)$ is the averaged reliability, i.e., the integrated reliability divided by the number of accumulated frames. p_u is the uniform probability distribution. λ_r is a parameter to modulate the effect of the reliability, and we set λ_r as 1.5 in the experiments as the maximum value of $W(v)$ was about 0.6.

Pairwise potential ψ_p The pairwise potential $\psi_p(x_i, x_j)$ consists of three bilateral kernels:

$$\begin{aligned} \psi_p(x_i, x_j) &= \mu(x_i, x_j) \left(w_a k_a(x_i, x_j) + w_n k_n(x_i, x_j) + w_s k_s(x_i, x_j) \right), \\ \mu(x_i, x_j) &= \begin{cases} 1, & \text{if } x_i \neq x_j \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{5.7}$$

where μ introduces a high penalty to differently labeled but similar neighboring nodes. The similarity between nodes are defined by following three kernels.

The appearance kernel $k_a(x_i, x_j)$ lets geometrically nearby vertices with similar appearances have the high similarity:

$$k_a(x_i, x_j) = \exp\Big(- \frac{|p_i - p_j|^2}{2\theta_p^2} - \frac{|c_i - c_j|^2}{2\theta_c^2} \Big), \tag{5.8}$$

where p_i and c_i are 3D vertex position and color of vertex v_i, respectively. Similarly, the surface smoothness kernel $k_n(x_i, x_j)$ enforces locally consistent predictions on smooth surfaces by taking into account the positions and surface normals:

$$k_n(x_i, x_j) = \exp\Big(- \frac{|p_i - p_j|^2}{2\theta_p^2} - \frac{|n_i - n_j|^2}{2\theta_n^2} \Big), \tag{5.9}$$

where n_i is the surface normal vector of vertex v_i.

Lastly we formulate the semantic similarity kernel $k_s(x_i, x_j)$ between two vertices utilizing the confusion matrix of 2D semantic segmentation network. A confusion matrix encodes how much each object class can be confused with other classes. For a vertex that has the maximum integrated confidence value on class k, we interpret the k-th row of the confusion matrix as a semantic feature.

$$k_s(x_i, x_j) = \exp\Big(- \frac{|p_i - p_j|^2}{2\theta_p^2} - \frac{|s_i - s_j|^2}{2\theta_s^2} \Big), \tag{5.10}$$

$$s_i = \mathcal{M}_{(k,:)}, \qquad \text{(the k-th row of matrix } \mathcal{M})$$

$$k = \text{argmax}_m C^m(v_i), \tag{5.11}$$

where $\mathcal{M}$ is the confusion matrix and $C^m(v_i)$ is the confidence of vertex v_i on object class m. By incorporating the confusion matrix to define the similarity of the confidences of two vertices, instead of directly comparing the classes with the maximum confidence values, different but similar classes (e.g., table and desk) can have higher values in the semantic similarity kernel k_s, encouraging the corresponding nodes to have the same label finally by CRF regularization even though their integrated confidences are not accurate. Similar to the depth-based accuracy weight in Section 5.5.1, we evaluate RDFNet [89] on ScanNet validation set to obtain the confusion matrix $\mathcal{M}$.

The Gibbs energy $E(\mathbf{x})$ can be efficiently minimized using a mean field approximation algorithm proposed by Krähenbühl and Koltun [90]. The algorithm inferences the result in linear time in the number of nodes. Therefore, despite the fact that our dense surface reconstruction process produces a mesh with millions of vertices, the CRF-based mesh segmentation only takes a few seconds.

In the following experiments, based on previous literatures [79, 90, 78], we use Gaussian parameters $\theta_p = 0.1$, $\theta_c = 0.1$, $\theta_n = 0.1$ and $\theta_s = 0.3$. We also empirically set the balance parameters as $w_a = 10$, $w_n = 10$ and $w_s = 3$.

5.6 Experimental Results

5.6.1 Experimental setting

We conducted experiments to evaluate our method using ScanNet dataset [72], which is a RGB-D stream collection of large scenes captured by Structure sensor [96] and semantically annotated by crowd workers. All the experiments were performed on a PC with an Intel i7-6700K 4.0GHz CPU, 32GB RAM and NVidia GTX 1080ti GPU. The publicly available implementations of dense surface reconstruction [20], 2D semantic segmentation [89], and CRF inference [90] were used with proper modifications to build the proposed framework. For RDFNet, we use the single-scale prediction rather than multiple-scale ensemble for efficiency.

Computation time Dense 3D reconstruction and volumetric fusion of 20 class semantic labels run in near real-time, but 2D segmentation takes about 0.3s per frame, preventing our framework from achieving real-time performance. In our current implementation using a single GPU, both components cannot be run at the same time due to the limited resource of the GPU. Mesh extraction and CRF regularization take about tens of seconds for a scene, which depends on the size of the reconstructed mesh.

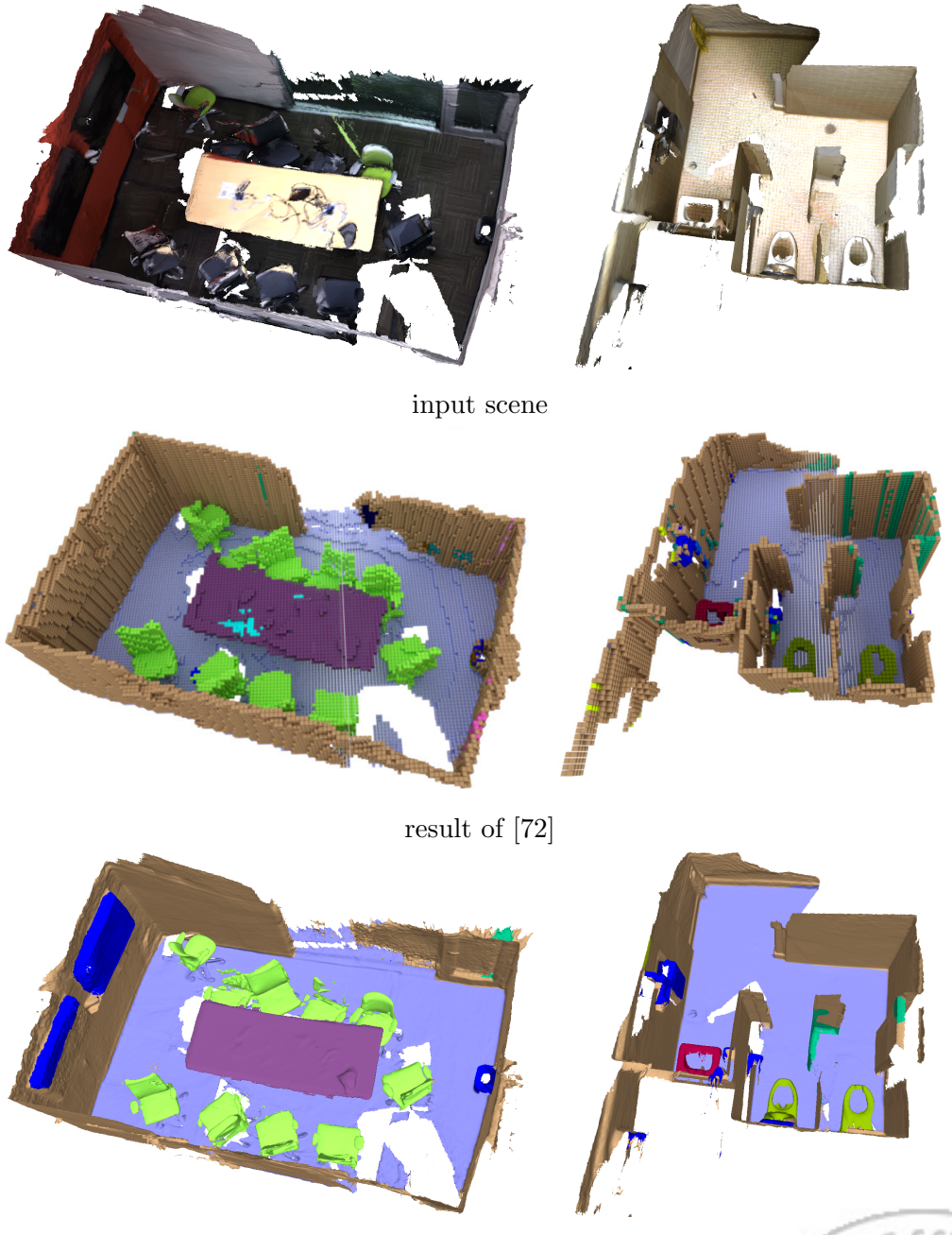
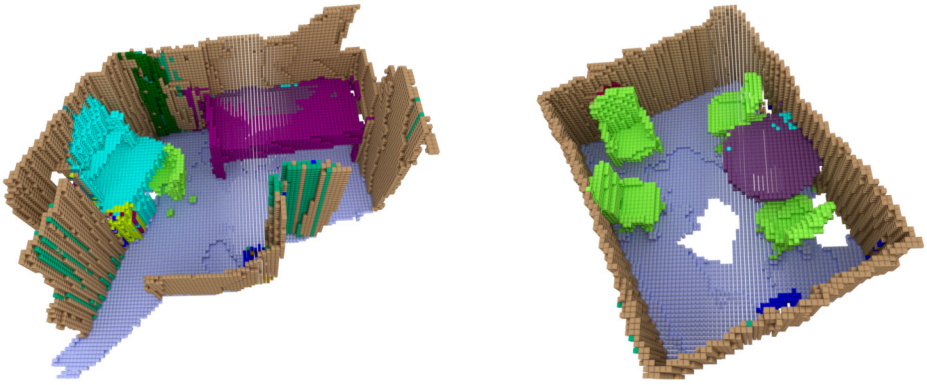
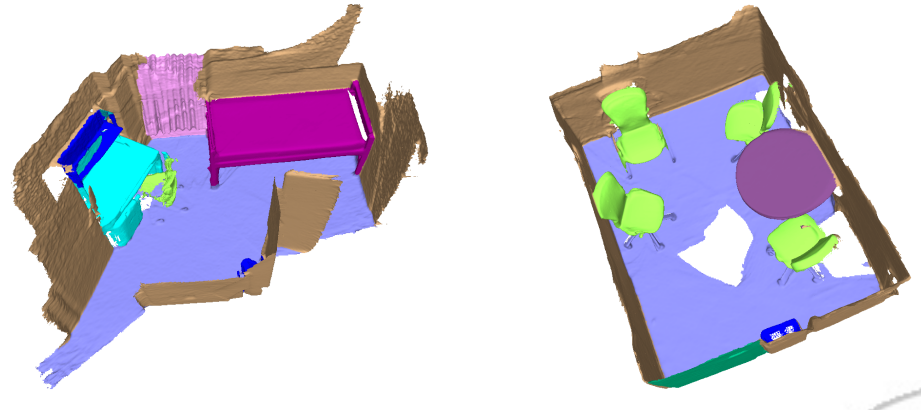


Figure 5.6: Comparison with the results of Dai et al. [72]. We used same false coloring as [72].

input scene



result of [72]



our result

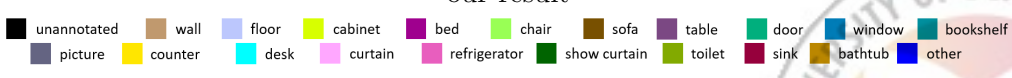
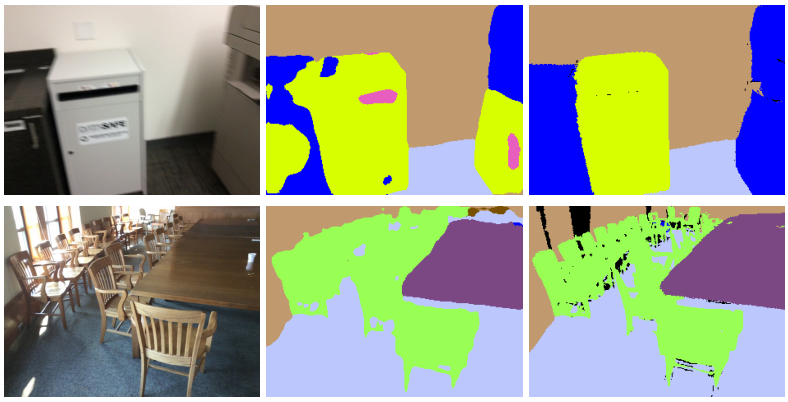


Figure 5.7: Comparison with the results of Dai et al. [72]. We used same false coloring as [72].



(a) color image (b) single-image result (c) projected result

Figure 5.8: Comparison between single-image semantic segmentation results and the projected results of our semantically segmented 3D meshes. Projection of missing geometry may introduce unlabeled pixels (black color).

5.6.2 Qualitative evaluation of segmentation results

Figs. 5.6-5.7 and 5.10 show the final results of the proposed semantic reconstruction framework. For a given RGB-D stream, our framework produces a high-quality dense 3D triangle mesh with semantically labeled vertices. As shown in the figures, the mesh is precisely segmented along the object class boundaries, such as between floor and wall.

Visual comparison with voxel-based method Dai et al. [72] proposed a 3D CNN-based semantic labeling algorithm using a low-resolution volume as the input and producing voxel labels as the output. Here we visually compare the labeling results of RGB-D streams with [72]. As shown in Fig. 5.6-5.7, our segmentation results show comparable quality in the perspective of labeling large structural object classes, such as floor, wall, and bed. Moreover, as our framework works on a dense triangle mesh, our results can distinguish the labels of small-scale objects that cannot be adequately handled by low-resolution volumes.

(a) input scene (b) naive integration (c) adaptive integration, no CRF (d) final result

Figure 5.9: Experiments on the effects of two major components in our framework.

Projected semantic segmentation images Since we have camera parameters of the input frames, we can project the semantically segmented mesh using the parameters to obtain 2D segmentation results. As shown in Fig. 5.8b, outputs of the 2D semantic segmentation network (fine-tuned RDFNet [89]) contain large mislabeled regions as the network only considers local contents of a given frame. In contrast, our projected segmentation results show remarkably enhanced quality and details thanks to the multiple-frame integration and CRF regularization process (Fig. 5.8c).

Component analysis We proposed adaptive integration and CRF regularization to exploit the characteristics of RDFNet and the global context of the scene. To evaluate the effects of these components, we test the framework with and without them. Fig. 5.9 shows the results of four cases of component analysis where each component is used or not. As expected, adaptive integration reduces mislabeled vertices around the boundaries of objects, such as chair and table, and CRF regularization drastically reduces small noisy labels.

5.6.3 Quantitative experiments

In addition to visual comparison, we quantitatively evaluate our semantic segmentation results using the semantic annotation of vertices in ScanNet

dataset [72]. For the test, we used ScanNet test dataset consists of 312 RGB-D streams in total.

Evaluation on ScanNet dataset Since 3D reconstruction produces different meshes according to the parameters, the ground truth mesh in ScanNet dataset and our result mesh may not share the same geometric structure (i.e., vertex and edge connectivity). Consequently, we cannot directly compare the vertex labels of our result mesh to the vertex labels in ScanNet dataset.

To evaluate the segmentation results independently of the mesh structures, we treat a result mesh as a point cloud by ignoring the edges. Following the approach described in [77], we voxelize the point cloud and the ground truth by merging the points in each voxel, where majority voting is used to decide the label of a voxel when several points belong to the voxel. Then we measure the global accuracy of the entire dataset by comparing the result voxel labels with the ground truths for all scenes. If a voxel is empty or not annotated, it is not counted in the evaluation.

Table 5.1 shows that our framework could achieve global accuracy of 80.5% over the 312 test scenes. It also shows accuracies of different settings of our framework and comparison to the voxel-based semantic labeling of [72]. Note that [72] voxelizes the scene into a low-resolution sparse grid and labels each grid cell. In contrast, we densely label each vertex with a semantic class, which is a relatively hard problem setting. In that sense, direct comparison between our results and [72] with the voxelization would not be fair although still our method shows a higher accuracy. Our adaptive integration and CRF regularization mainly address the mislabeled and noisy vertices around object boundaries, as shown in Fig. 5.9. These improvements are clear in terms of visual quality but may not introduce significant changes on the global accuracy.

Configurations	Accuracy
Voxel-based labeling [72]	73.00%
Naive integration and without CRF	79.02%
Adaptive integration and without CRF	79.28%
Naive integration with CRF	79.79%
Adaptive integration and with CRF	79.86%

Table 5.1: Global accuracy of semantic segmentation on ScanNet. For the naive integration, we use a uniform weight for per-pixel reliability W_{F_t}.

Comparison with point-based methods RSNet [77] mentioned that the global accuracy is not enough for evaluating segmentation results on highly unbalanced ScanNet dataset. For better evaluation of our results, we measured the mean intersection over union (mIOU) and mean accuracy (mAcc), as in [75, 76, 77]. Table 5.2 shows our method achieved the state-of-the-art results on ScanNet dataset. Compared to RSNet with RGB information, ours improves mIOU and mAcc by 2.84% and 15.30%, respectively. Table 5.2 also shows IOU of each category.

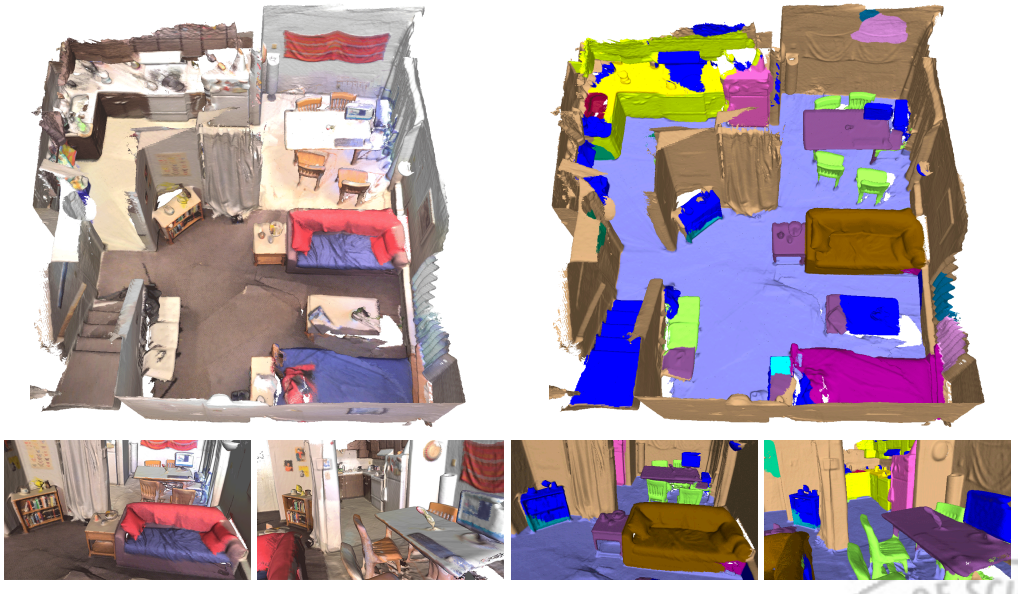
Evaluation on projected 2D segmentation We also evaluate the performance of our semantic integration by comparing the results against the single image 2D semantic segmentation results, which have been used for volumetric fusion. We measure three types of accuracy metrics (pixel accuracy, mean accuracy [83], and mean IOU [97]) on 53K semantic segmentation results in total by projecting every 10th frame in the 312 ScanNet test scenes. The results in Table 5.3 are matched with the visual comparisons in Fig. 5.8, and our semantic reconstruction improves the segmentation accuracy over the original 2D semantic segmentation.

	PointNet [75]	PointNet++ [76]	RSNet [77]	RSNet w/ RGB [77]	Ours
mIOU	14.69	34.26	39.35	41.16	**44.00**
mAcc	19.90	43.77	48.37	50.34	**65.64**
wall	69.44	77.48	79.23	79.38	66.90
floor	88.59	92.50	94.10	94.21	80.67
cabinet	4.99	23.81	31.29	30.06	31.76
bed	17.96	51.32	55.95	53.09	52.66
chair	35.93	64.55	64.99	63.65	64.01
sofa	32.79	52.27	55.41	51.06	58.39
table	32.78	46.60	51.04	48.67	51.64
door	0.00	2.02	3.00	15.32	30.93
window	0.00	3.56	8.75	15.67	21.10
bookshelf	3.18	52.93	53.02	53.67	31.21
picture	0.00	0.00	0.95	4.30	7.38
counter	5.09	20.04	22.72	20.90	24.07
desk	2.63	12.69	34.53	35.27	26.18
curtain	0.00	32.97	6.78	8.30	30.36
refridg	0.00	18.51	37.90	39.76	56.34
shower	0.00	27.43	29.92	24.36	23.63
toilet	0.00	31.37	54.16	63.20	73.37
sink	0.00	30.23	34.84	41.00	46.26
bathtub	0.17	42.72	49.38	60.37	69.74
others	0.13	2.20	18.98	20.98	33.33

Table 5.2: Quantitative results on ScanNet dataset. We report mean IOU and mean Accuracy of all categories as well as IOU for each category. Measurements of previous methods are from [77].

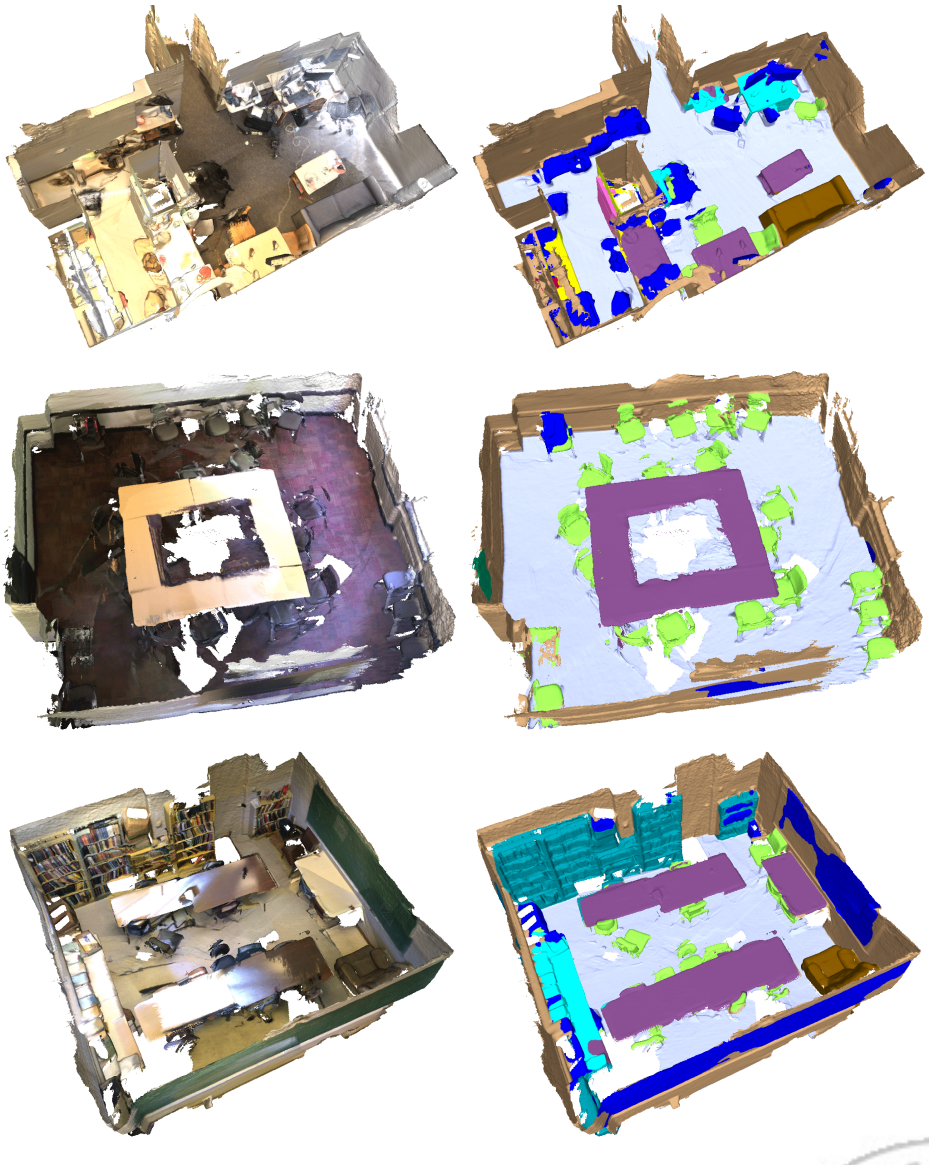
Method	pixel acc.	mAcc.	mIOU
single image segmentation (original) [89]	60.44	47.32	29.34
single image segmentation (fine-tuned on ScanNet)	73.55	59.82	45.60
projected segmentation	77.18	63.20	50.69

Table 5.3: Segmentation accuracy of 2D segmentation results and our projected results measured using projected images of ScanNet [72] test scenes.



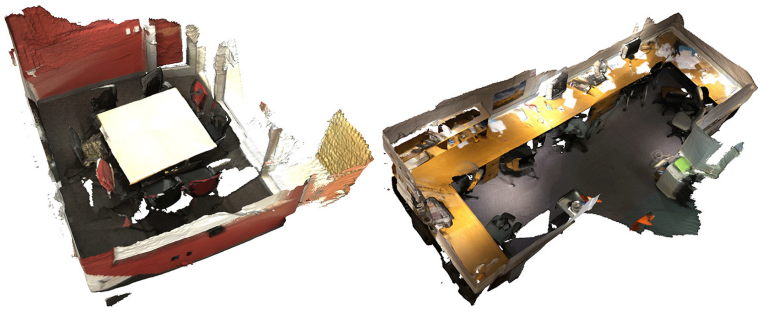
reconstructed scene our segmentation result

Figure 5.10: Semantic segmentation results of large-scale complex scenes reconstructed using thousands of RGB-D frames.

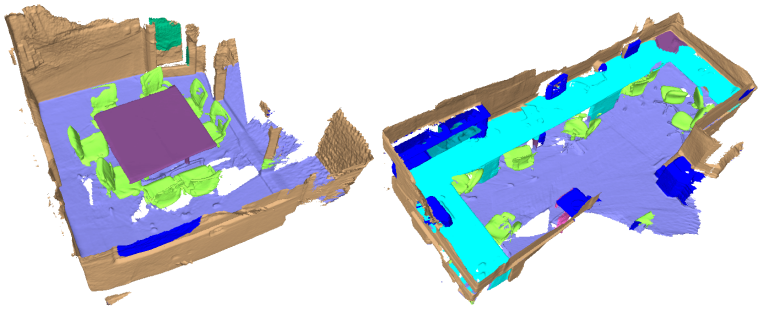


input scene our segmentation result

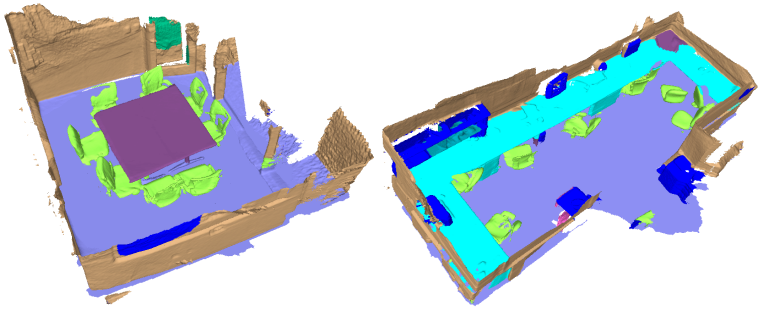
Figure 5.11: Semantic segmentation results of additional reconstructed scenes.



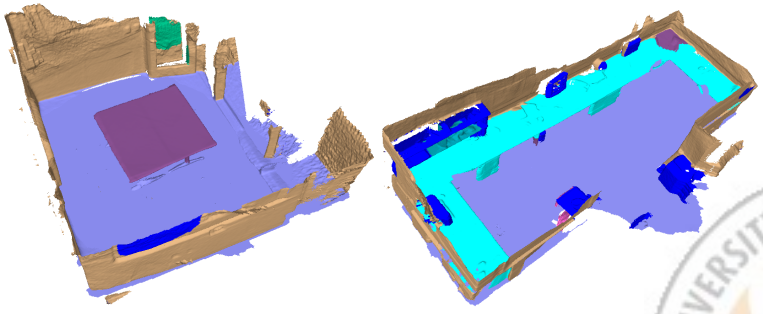
(a) original scene



(b) manipulated scene



(c) completed scene



(d) completed & manipulated scene

Figure 5.12: Geometric hole-filling & scene manipulation (removing the chairs).

5.6.4 3D Scene completion and manipulation

As applications of our 3D semantic segmentation, we demonstrate 3D scene completion and manipulation. Single surface reconstruction using a RGB-D stream cannot build a complete geometry of the scene due to uncaptured regions. For example, a floor region under the desk or bed cannot be reconstructed as it is not visible from any input frame. This incomplete geometry could limit the usage of the reconstructed scene model in various applications, such as virtual reality and interior redesign. For example, removing or moving objects on the floor would reveal the uncaptured regions, which we call geometric holes.

Semantic-aware scene completion Geometric holes of the reconstructed mesh can be easily filled using semantic information. Dense 3D reconstruction algorithms such as BundleFusion [20] and VoxelHashing [17] generate a single huge connected geometry without any semantic information, and it is not straightforward to detect scene structures, such as wall, floor, and ceiling. In contrast, our method generates a semantically segmented mesh, and we can easily estimate the structures by grouping vertices with the same semantic labels.

For example, we can fill the geometric holes on the floor with a few simple steps. First, we fit a 3D plane to the floor by performing RANSAC [95] on the floor vertices. We subdivide the estimated plane into a 2D grid and project all vertices with floor and wall class labels onto the plane, finding unoccupied grid cells, which correspond to geometry holes. We can then extract the hole boundary using a simple contour tracing algorithm [98]. Finally, re-triangulating the grid cells inside the hole boundary gives us a clean, hole-filled floor mesh. As shown in Fig. 5.12c, our geometric hole-filling algorithm effectively recovers the missing parts on the floor, which has not been captured in the reconstruction process.

3D Scene manipulation As we have determined the semantic labels of all the vertex, we can easily divide the entire reconstructed mesh into separate meshes for different object classes. Then we can manipulate the individual object meshes independently from others, e.g., by applying 3D transforms, as shown in Fig. 5.12d. Note that in the example, objects on the floor can be freely moved without revealing any holes as we have already restored the complete floor plane in our semantic-aware scene completion step.

5.7 Discussion

This paper introduced a novel framework that automatically generates a semantically segmented triangular mesh from a RGB-D video. Our method exploits the recent successes of deep neural networks on semantic segmentation of images by adaptively integrating 2D semantic predictions through volumetric fusion. In addition, our CRF-based semantic label regularization produces a more robust segmentation result by incorporating global scene context using geometric and photometric information.

Our method does not require a labeled 3D training dataset, which would be harder to obtain than a 2D dataset. In addition, advances of 2D semantic segmentation can be readily incorporated for improving 3D segmentation in our framework. For example, 3D instance segmentation could be made possible with our approach as high quality 2D instance segmentation becomes available.

Limitation and future work Our framework contains 3D reconstruction and RGB-D image segmentation, and any failure on each process will lower the quality of the results. Especially, camera tracking failure or drifting during the reconstruction may introduce erroneous results. Noisy and inaccurate results of 2D semantic segmentation are less critical as our method compensates the errors by

integrating segmentation results from multiple frames (Fig. 5.8. Our future work includes real-time semantic reconstruction of 3D scenes that can provide integrated semantic information, as well as geometry and color, of the reconstructed scene during the capturing process, which would need multi-GPU implementation. It would be also interesting to investigate challenging problems in computer graphics such as semantic modeling [99, 100] by incorporating the semantically segmented meshes.

VI. Conclusion and Future Work

6.1 Summary

In this thesis, we have proposed two novel color reconstruction and one semantic reconstruction techniques to generate a ready-to-use 3D model of the indoor scene for computer graphics applications.

In Chapter 3, we proposed a novel intrinsic image decomposition method which exploits the surface normal constraint from an RGB-D image. The proposed method can handle the oscillating texture patterns in an input image as our new image model explicitly considering them. As a result, our method can produce superior results to existing approaches. Besides, the decomposed reflectance image can be used in further color reconstruction process such as the rendering of reconstructed scenes with modified lighting condition.

In Chapter 4, we presented a novel texture map generation for 3D reconstructed scenes. To represent the color information of 3D reconstruction, the proposed method uses a texture mapping with simplified mesh instead of millions of vertices color. Our GPU-based texture coordinates optimization produces photometrically consistent high-quality texture map in few tens of seconds for a large 3D model such as a whole room. Moreover, by combining with intrinsic image decomposition in Chapter 3, we also can generate a reflectance texture map which can be rendered without being disturbed by original lighting condition.

In Chapter 5, we presented a semantic segmentation method for 3D reconstructed scenes. To precisely reconstruct the semantically segmented 3D reconstructed triangular meshes, the proposed method uses a volumetric semantic fusion algorithm to integrate multiple 2D CNN-based semantic segmentation results

of RGB-D stream. Our method can be seamlessly integrated into the original 3D reconstruction process that uses a volumetric representation for the geometry without introducing any additional data structure. We demonstrated that our method outperforms existing state-of-the-art 3D semantic segmentation methods based on point cloud or volumetric CNN.

6.2 Limitation and Future Work

Intrinsic image decomposition Because our decomposition method is developed for a single RGB-D image, the temporal consistency among the multiple frames cannot be held if we independently apply the method to the RGB-D stream for a 3D reconstructed scene. As a future work, we would like to extend our intrinsic image decomposition to consider the global context of the reconstructed scene similar to our recent researches on the texture generation and semantic reconstruction.

Texture map generation The proposed texture map generation method optimizes the texture coordinates to only maximize the photometric consistency of the generated texture map. So the generated texture map may be misaligned with the corresponding geometry. Optimizing the texture map considering the geometric and photometric consistency at the same time would be an interesting future work.

Semantic reconstruction Since the semantic segmentation method only predicts the object class but not the individual instance, the proposed method cannot separate the reconstructed mesh to the individual objects. We plan to extend our method to segment the reconstructed scene instance-wise incorporating the CNN-based 2D instance segmentation techniques to our framework.

요 약 문

최근 3D 복원 기술의 발전에도 불구하고 대부분의 연구는 기하 정보만을 복원하는 것에 집중해왔다. 많은 컴퓨터 그래픽스 어플리케이션들이 기하 정보 외에도 환경의 조명, 물체 표면의 재질, 의미론적 분류 등 다양한 보조 정보를 요구하지만 기존의 3D 복원 기술은 이러한 보조 정보 복원을 고려하지 않는다는 한계점이 있었다. 그럼에도 불구하고 이러한 보조 정보를 복원하는 것은 최근 각광받고 있는 가상현실이나 증강현실 등에 풍부한 사용자 경험을 제공해준다는 점에서 연구할만한 가치가 있다. 본 학위논문에서는 기하 정보 이외의 보조 정보를 복원하는 세 가지 기술인 본질 영상 분리, 3D 복원을 위한 텍스쳐 맵 생성, 3D 의미론적 복원을 제안한다.

첫 번째로 색상-깊이 영상을 입력으로 한 본질 영상 분리 시에 영상의 텍스쳐 무늬 분리를 통해 간단한 최적화로 좋은 성능을 얻을 수 있는 기술을 제안한다. 또한 노멀 맵을 바탕으로 한 기하 정보 기반의 새로운 음영 영상 제약 방법을 제시해 결과 음영 영상에서의 국소적, 전반적 일관성을 향상시키는 방법을 제안한다. 제안된 기술은 기존 기술에 비해 매우 뛰어난 성능을 보여준다.

두 번째로 3D 복원 모델을 위한 선명한 텍스쳐 맵 생성 기술을 새롭게 제안한다. 본 논문에서 제안하는 방식은 3D 복원을 위한 색상 표현 방식으로써 기존 정점의 색을 이용하던 방식을 벗어나 컴퓨터 그래픽스에서 주로 사용되는 텍스쳐 맵을 이용한다. 보다 선명한 텍스쳐 맵을 얻기 위해 시공간적 샘플링을 통해 텍스쳐 맵 생성에 사용되는 영상의 품질을 높이며, 색상 일관성을 최대화시키는 부 텍스쳐 맵을 GPU를 이용해 병렬적으로 최적화한다. 제안된 기술을 통해 방 하나 정도의 대규모 3D 복원 모델을 위한 텍스쳐 맵을 수 초 이내에 생성할 수 있다.

마지막으로 3D 복원 모델의 의미론적 분리를 위한 의미론적 복원 기술을 제안한

다. 본 논문에서 제시한 알고리즘은 3D 복원 도중 딥러닝 네트워크를 통해 입력 색상-깊이 프레임으로부터 의미론적 2D 맵을 얻고, 이를 3D 복원에 사용되는 균일한 복셀 격자에 누적시키는 의미론적 공간 병합 기술을 이용한다. 영상 신뢰도 기반의 가변적 병합과 조건부 임의 필드를 이용한 최적화를 통해 3D 복원 모델의 깔끔한 의미론적 분리가 가능하며, 기존 기술에 비해 정성적, 정량적으로 높은 성능을 보인다.

References

[1] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Transactions on Graphics (ToG)*, volume 25, pages 835–846. ACM, 2006.

[2] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 80–87. IEEE, 2009.

[3] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010.

[4] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew 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.

[5] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (TOG)*, 32(4):112, 2013.

[6] Matthias Niener, Michael Zollhfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013.

[7] Andrew W Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. European Conference on Computer Vision*, pages 311–326. Springer, 1998.

[8] Sudipta N Sinha, Philippos Mordohai, and Marc Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

[9] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[10] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528. IEEE, 2006.

[11] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1434–1441. IEEE, 2010.

[12] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *IEEE and ACM international symposium on Mixed and augmented reality (ISMAR)*, pages 225–234. IEEE, 2007.

[13] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011.

[14] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface

mapping and tracking. In *IEEE international symposium on Mixed and augmented reality (ISMAR)*, pages 127–136. IEEE, 2011.

[15] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

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

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

[18] Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. Elastic fragments for dense scene reconstruction. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 473–480, 2013.

[19] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015.

[20] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

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

[22] Jean-François Lalonde and Iain Matthews. Lighting estimation in outdoor image collections. In *Proc. International Conference on 3D Vision (3DV)*, volume 1, pages 131–138. IEEE, 2014.

[23] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[24] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Tinne Tuytelaars, and Luc Van Gool. What is around the camera. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, volume 17, 2017.

[25] Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. In *Proc. of SIGGRAPH*, pages 207–217. ACM, 1998.

[26] Pierre-Yves Laffont, Adrien Bousseau, Sylvain Paris, Frédo Durand, George Drettakis, et al. Coherent intrinsic images from photo collections. *ACM Transactions on Graphics*, 31(6), 2012.

[27] Erum Arif Khan, Erik Reinhard, Roland W Fleming, and Heinrich H Bülthoff. Image-based material editing. *ACM Transactions on Graphics*, 25(3):654–663, 2006.

[28] E. H. Land and J. J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61(1), 1971.

[29] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, 1978.

[30] B. V. Funt, M. S. Drew, and M. Brockington. Recovering shading from color images. In *Proc. European Conference on Computer Vision (ECCV)*, 1992.

[31] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel. A variational framework for retinex. *International Journal of Computer Vision*, 52:7–23, 2003.

[32] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[33] Y. Weiss. Deriving intrinsic images from image sequences. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2001.

[34] Y. Matsushita, S. Lin, S. B. Kang, and H.-Y. Shum. Estimating intrinsic images from image sequences with biased illumination. In *Proc. European Conference on Computer Vision (ECCV)*, 2004.

[35] P.-Y. Laffont, A. Bousseau, and G. Drettakis. Rich intrinsic image decomposition of outdoor scenes from multiple views. *IEEE Transactions on Visualization and Computer Graphics*, 19(2), 2013.

[36] K. J. Lee, Q. Zhao, X. Tong, M. Gong, S. Izadi, S. U. Lee, P. Tan, and S. Lin. Estimation of intrinsic image sequences from image+depth video. In *Proc. European Conference on Computer Vision (ECCV)*, 2012.

[37] J. T. Barron and J. Malik. Intrinsic scene properties from a single RGB-D image. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[38] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2013.

[39] A. Bousseau, S. Paris, and F. Durand. User-assisted intrinsic images. *ACM Transactions on Graphics*, 28(5), 2009.

[40] Vivek Kwatra, Mei Han, and Shengyang Dai. Shadow removal for aerial imagery by information theoretic intrinsic image analysis. In *International Conference on Computational Photography*, 2012.

[41] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[42] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[43] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 1998.

[44] A Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[45] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics*, 27(3):67:1–67:10, 2008.

[46] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via L0 gradient minimization. *ACM Transactions on Graphics*, 30(6):174:1–174:12, 2011.

[47] Kartic Subr, Cyril Soler, and Frédo Durand. Edge-preserving multiscale image decomposition based on local extrema. *ACM Transactions on Graphics*, 28(5):147:1–147:9, 2009.

[48] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics*, 31(6):139:1–139:10, 2012.

[49] Levent Karacan, Erkut Erdem, and Aykut Erdem. Structure-preserving image smoothing via region covariances. *ACM Transactions on Graphics*, 32(6):176:1–176:11, 2013.

[50] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *Proc. European Conference on Computer Vision (ECCV)*, pages 589–600. Springer Berlin Heidelberg, 2006.

[51] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[52] L. Shen and C. Yeo. Intrinsic images decomposition using a local and global sparse representation of reflectance. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[53] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7), 2012.

[54] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Proc. European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, 2012.

[55] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *Proc. European Conference on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[56] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2009.

[57] Henry Roth and Marsette Vona. Moving volume kinectfusion. In *BMVC*, pages 1–11, 2012.

[58] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust tracking for real-time dense rgb-d mapping with kintinuous. 2012.

[59] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[60] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3001–3008. IEEE, 2011.

[61] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[62] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer*

graphics and interactive techniques, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[63] Frédérique Crété-Roffet, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: Perception and estimation with a new no-reference perceptual blur metric. In *SPIE Electronic Imaging Symposium Conf Human Vision and Electronic Imaging*, volume 12, pages EI–6492, 2007.

[64] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, 2002.

[65] Michiel Hazewinkel, editor. *Encyclopaedia of Mathematics (set)*. Springer Netherlands, 1994.

[66] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. MeshLab: an Open-Source 3D Mesh Processing System. *Ercim News*, 2008, 2008.

[67] Blender Foundation. Blender.

[68] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pages 1495–1504. Wiley Online Library, 2008.

[69] Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)*, 34(4):70, 2015.

[70] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[71] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[72] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[73] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016.

[74] Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. PiGraphs: Learning Interaction Snapshots from Observations. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[75] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[76] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[77] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018.

[78] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2017.

[79] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3D semantic mapping of indoor scenes from rgb-d images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638. IEEE, 2014.

[80] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017.

[81] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[82] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[83] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[84] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[85] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[86] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[87] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. European Conference on Computer Vision*, 2012.

[88] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision*, pages 213–228. Springer, 2016.

[89] Seong-Jin Park, Ki-Sang Hong, and Seungyong Lee. Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[90] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[91] Jing Huang and Suya You. Point cloud labeling using 3d convolutional neural network. In *International Conference on Pattern Recognition (ICPR)*, pages 2670–2675. IEEE, 2016.

[92] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[93] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 163–169. ACM, 1987.

[94] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[95] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987.

[96] Occipital. Occipital: The structure sensor, 2016.

[97] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[98] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[99] Kang Chen, Yukun Lai, Yu-Xin Wu, Ralph Robert Martin, and Shi-Min Hu. Automatic semantic modeling of indoor scenes from low-quality rgb-d

data using contextual information. *ACM Transactions on Graphics*, 33(6), 2014.

[100] Mingming Liu, Yanwen Guo, and Jun Wang. Indoor scene modeling from a single image using normal inference and edge features. *The Visual Computer*, 33(10):1227–1240, 2017.

Acknowledgements

먼저 지난 7년 동안 많은 가르침으로 저를 이끌어주신 지도교수 이승용 교수님께 감사 드립니다. 박사학위과정이라는 멀고 험한 길을 가면서도 목적지를 잃지 않고 똑바로 나아갈 수 있었던 것은 모두 교수님 덕분이었습니다. 앞으로 사회의 어디에서든 교수님의 가르침을 잊지 않고 하루하루 발전하는 삶을 살겠습니다.

바쁘신 와중에 소중한 시간을 저의 부족한 학위논문 심사에 써주신 홍기상 교수님, 조민수 교수님, 곽수하 교수님, 윤국진 교수님께 감사드립니다. 특히 먼 포항까지 두 번이나 오셔서 저를 위한 시간을 내어 주신 윤국진 교수님께 감사드립니다. 또한 여러 학과 수업을 통해 많은 가르침을 주신 포항공과대학교 컴퓨터공학과의 교수님들께도 감사 드립니다. 중국 MSRA의 인턴 기간 동안 멘토로써 많은 영감을 준 Xin Tong 박사에게도 감사를 전합니다.

지난 7년의 통합과정 기간동안 저와 함께했던 많은 선후배님과 동료들에게 모두 감사의 말씀을 전합니다. 부족한 후배를 잘 챙겨주셨던 현준이형, 성현이형, 용진이형, 호진이형, 철훈이형, 치영이형, 대훈이형과 민정누나, 후배로 들어왔지만 먼저 사회로 나가 인생의 선배가 된 형석, 연규, 해준, 재환이, 최근에 졸업해 갓 사회에 첫발을 내딛은 은빈이와 범석이, 이제 나와 함께 졸업할 재필이, 이제 랩을 책임져야 할 손형석이, 머리는 좋은데 잘 안쓰는 진웅이, 누구보다 연구실에 오래 있는 준건이, 연구와 가정 둘 다 놓치지 않는 준용이, 나 다음으로 졸업해야할 건희와 Jake, 예의바른데 귀여운 종협이, 어려운 3D 연구를 맡아 잘 해내고 있는 효민이, 중국에서 폐관수련중인 유철이와, 그동안 연구실의 살림을 책임져줬던 현진, 현이, 혜은, 지혜누나까지 정말 많은 분들의 도움에 항상 감사 드립니다.

또한 마지막으로, 무엇보다 저를 낳고 지금까지 길러주신 부모님께 감사합니다. 긴 시간 포항에서 공부하며 자주 찾아뵙지도 못하고 걱정만 끼친 저를 믿고 지지해주신 부모님의 믿음에 보답하겠습니다.

지금까지 인생의 1/3이 넘는 긴 포항 생활이 늘 행복하고 즐거운 시간이었던 것은 늘 제 곁을 지켜주고 응원해준 소중한 인연 덕분이라 생각니다. 그동안 포항에서 지내면서 배운 많은 것들과 여러 사람으로부터 받은 많은 도움을 바탕으로 훌륭한 사람이 되겠습니다. 모든 분들께 다시 한번 감사드립니다.

2018년 12월 전준호 드림

Curriculum Vitae

Name : Junho Jeon

Education

2008.3 – 2012.2 Department of Computer Science and Engineering, Pohang University of Science and Technology (B.S.)

2012.3 – 2019.2 Department of Computer Science and Engineering, Pohang University of Science and Technology (Ph.D.)

Experience

2012. 9. – 2013. 2. Research Intern at Microsoft Research Asia
Internet Graphics Group. Beijing, China