Planar Abstraction and Inverse Rendering of 3D Indoor Environments

Young Min Kim, Sangwoo Ryu, and Ig-Jae Kim

Abstract—Scanning and acquiring a 3D indoor environment suffers from complex occlusions and misalignment errors. The reconstruction obtained from an RGB-D scanner contains holes in geometry and ghosting in texture. These are easily noticeable and cannot be considered as visually compelling VR content without further processing. On the other hand, the well-known Manhattan World priors successfully recreate relatively simple structures. In this article, we would like to push the limit of planar representation in indoor environments. Given an initial 3D reconstruction captured by an RGB-D sensor, we use planes not only to represent the environment geometrically but also to solve an inverse rendering problem considering texture and light. The complex process of shape inference and intrinsic imaging is greatly simplified with the help of detected planes and yet produces a realistic 3D indoor environment. The generated content can adequately represent the spatial arrangements for various AR/VR applications and can be readily composited with virtual objects possessing plausible lighting and texture.

Index Terms—3D content creation, indoor modeling, texture generation, inverse rendering

1 Introduction

A realistic 3D environment has extensive possible applications. The immediate usage would be visualizing 3D content for commercial solutions indoors, namely real estate or interior designs of homes, offices, or hotel rooms. Realistic VR contents could even be a solution for treating mental issues including Alzheimer diseases [1], [2], [3], or for producing training data for machine-learning tasks with physically based rendering [4] and a simulator [5]. The reconstructed environment can ultimately perform a higher level of understanding and manipulations of everyday environments when equipped with semantics and intelligent agents.

Recent advances of 3D scanning technology allow very accurate reconstruction of 3D objects in a controlled setting with one or more cues of stereo reconstruction [6], shading [7], silhouette [8], or depth sensors [9], [10]. However, conventional 3D reconstruction technology cannot simply be extended into un-controlled large-scale indoor environments. This is because the collected data suffers from clutter, and there is often insufficient overlap between the camera frames [11].

The representation of indoor environments is usually based on either 3D reconstruction or image-based rendering. The former has easily noticeable artifacts due to holes, misalignment of texture (or depth boundaries), changed exposures between different frames, and extensive smoothing of small geometry. On the other hand, the latter shows a photo-realistic texture that is visually compelling when the viewpoint is similar to that from which the original image was captured. The rendering quality, however, degrades significantly with substantial changes in viewpoint, especially for geometrically complex objects. Image-based rendering does not have an underlying geometry that could enable manipulations such as changes of object poses, lighting or materials.

In this paper, we present the power of plane-based abstraction to better represent the majority of static geometric structures in indoor environments and complete the occluded parts (Fig. 1). Because planar-based abstraction has already proved its effectiveness in large-scale urban modeling or Manhattan-world based indoor modeling [12], [13], [14], [15]; indoor environments are well represented and separated by planar structures, mainly walls, floors, and desktops.

We demonstrate that detected planes can play an important role not only for geometric completeness, but also can be extended for texturing, lighting, and registration. Our intuition comes from the fact that computer graphics have long benefited from billboard-based abstraction with bump maps or texture to render high-quality images quickly. Similarly, we can use the underlying base plane primitives with better texture to create illusions of detailed reconstruction, and further extract intrinsic components. The underlying assumption is that the bare-bones geometry of the indoor environment would be surrounded completely by a planar structure (walls, ceiling, and floor), with homogeneous texture.

Starting from 3D-scanned incomplete geometry of the indoor environment, we first detect major planes within the scene and optimize over registration and exposure terms of individual frames observing each plane to have a consistent

- *Y. M. Kim is with the Seoul National University, Seoul, South Korea. E-mail: youngmin.kim@snu.ac.kr.*
- *S. Ryu is with the Pohang University of Science and Technology (POSTECH), Pohang-si, Gyeongsangbuk-do, South Korea. E-mail: rswoo@postech.ac.kr.*
- *I.-J. Kim is with the Korea Institute of Science and Technology (KIST), Seoul, Republic of Korea. E-mail: drjay@kist.re.kr.*

Manuscript received 21 Feb. 2019; revised 30 Nov. 2019; accepted 6 Dec. 2019. Date of publication 19 Dec. 2019; date of current version 29 Apr. 2021. (Corresponding author: Young Min Kim.)
Recommended for acceptance by A. Ghosh.
Digital Object Identifier no. 10.1109/TVCG.2019.2960776

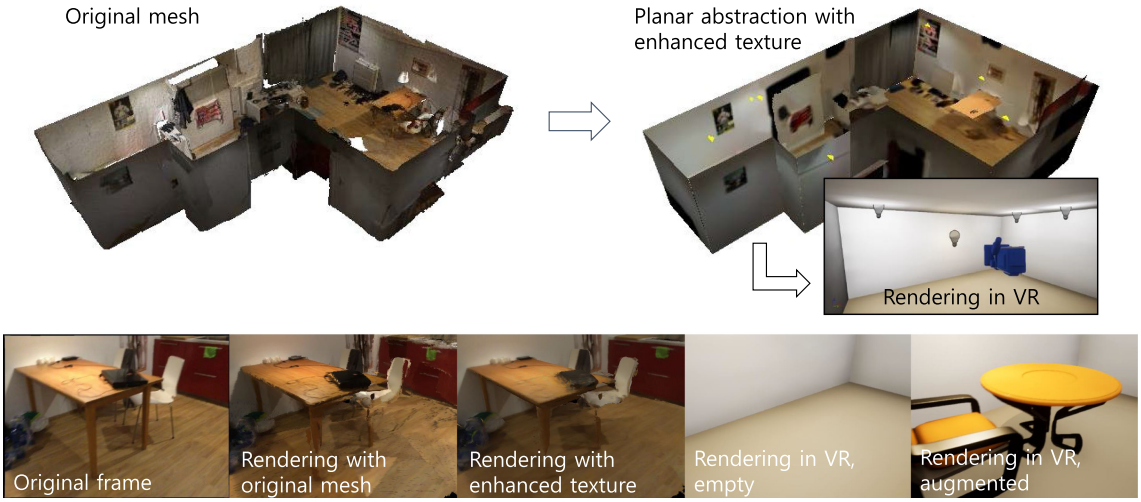


Fig. 1. Sample modeling results using our pipeline. We present a system that can generate corresponding 3D models from multiple photographs provided the topology of the object category. The topological prior says how individual parts of an object category are connected and compose an object. For example, a teapot (left) has one straight axis in the middle and two curved axes, and a vase (right) has a straight axis and can optionally have a pair of symmetric axes. The sampled input photographs and output textured models are presented on each side of the arrows.

texture. During the process, the texture is decomposed into the homogeneous, flat background and occluding foreground with variation and details. The flat geometry and background texture of the planes are then interpolated and extrapolated to fill the inevitable holes and to complete the room structure, similar to [16].

In addition to the geometry estimate, corresponding material and texture information is necessary to augment the model with realistic rendering in an AR/VR scenario [17], [18], [19]. We further exploit the assumption to deduce the direct and indirect lighting and background materials jointly. This demonstrates the extended potential for large-scale virtual applications of 3D indoor environments with plausible rendering. Even in a confined space, indoor lighting varies for different locations due to the effects of light fixtures, windows, and shadows of complex objects. In previous work, these effects are often ignored, and large-scale lights are simply modeled with environment maps or directional lighting [20], [21], [22], [23]. On the other hand, we directly locate the direct light sources and reflect the effects of different locations from them in addition to environmental lighting.

We offer three main contributions:

- converting 3D reconstruction of an indoor environment into an abstract and compact representation of a realistic indoor environment based on planar primitives;
- decomposing the created content using an inverse rendering pipeline, and making it readily available in the form of physically-based rendering with joint analysis of shape, texture, and lighting;
- extracting basic semantics of foreground/background segmentation utilizing recovered geometry and texture.

We demonstrate that our representation creates realistic visualizations for various indoor spaces and at the same time decomposes the scene semantically (walls, floors, desktops, and objects) and intrinsically (geometry, lighting, and texture). The new form of visualization is compared with other recent representations of 3D indoor environments [16], [24], [25].

2 Related Work

Reconstructing and visualizing large-scale indoor environments have a variety of practical applications. Some examples range from generating AR/VR systems to incorporating an intelligent system into the geometry and semantics of a real-world environment or augmenting data for machine learning [4]. Acquiring large-scale uncontrolled indoor data, however, is challenging. Everyday environments are highly complex and exhibit occlusions with various objects. There exist static furniture and non-static objects, and often the accessibility of the viewpoints is limited to the remaining open space.

Three major approaches exist by which to represent a real-world indoor environment. First, the volumetric fusion of large-scale environments converts the accumulated depth measurement into a dense mesh [25] that could be further used for useful high-level tasks such as semantic segmentation [24], [26] or to simulate activities [5]. This representation creates the most thorough geometry of the actual environment. The texture is usually created by projecting RGB frames onto the measured depth data, but there are possible misalignment issues that create ghosting artifacts. The misalignment comes from either imprecise geometry of the reconstruction method that smooths out small details by accumulation or from calibration errors between multiple frames with insufficient overlap. In addition to texture errors, there exist inevitable holes in measurements when scanning large environments.

The second approach to representation is image-based rendering ([11], [27] and the references therein), which does not have the texture misalignment problem. Instead, image-based rendering techniques create as crisp an image as possible by deforming the existing image frames based on rough estimates of the underlying geometry. The rough geometry estimate only serves as a guide to warp images to create a realistic visualization. While the reflectance recovery or

augmentation becomes faithful [28] with accurate reconstruction and known illumination, visualization quality deteriorates if the viewpoint changes significantly from that in the measured frames. Without improved geometry, the representation can only be used for visualization and cannot be extended to other tasks such as free-navigation, scene editing, or semantic segmentation.

The third approach to representation uses a geometric proxy to represent the core elements of the indoor scenes. One of the most successful ways to represent large-scale indoor environments is by using a Manhattan-World prior to fill the holes in a stereo reconstruction [14], [15]. Planes are successfully used to create a plausible reconstruction of a building with limited inputs [12], [13], [29]. Huang *et al.* [16] used plane geometry and texture to fill holes and create plausible 3D content of indoor environments. Starting from the planar assumption to create a complete geometry as suggested in the literature mentioned above, our work incorporates light information to complete the rendering pipeline of scanned indoor environments.

In this paper, we present a plane-based approximation of indoor environments. We are not limited to mere geometric reconstruction, but also generate plausible texture and lighting parameters given the fitted planes. Previous literature for 3D object rendering and compositing extracted environmental light and material components from an image [17], [18], [19]. Similarly, we are solving intrinsic image decomposition [30], which is inherently under-constrained. We substitute shape and reflectance priors with the recovered planar scene geometry and reflectance of background walls, respectively. Our work is inspired by previous work that successfully exploited planar geometry for other tasks. Teichman *et al.* [31] used detected planes to solve jointly for SLAM and calibrate sensor intrinsics, while Jachnik *et al.* [32] solved for the entire light field of a small (book size) specular planar surface. Similarly, we jointly solve necessary calibration parameters and reflectance as well as light parameters. Zhang *et al.* [33] also used planar reconstruction to find reflectance of walls and light parameters. They only recovered indirect lighting as a cube map and 1-3 reflectance values of walls for the entire room; therefore, this cannot be applied in large-scale rooms with non-rectangular shapes. On the other hand, our pipeline embraces more general planar-geometries and recovers both direct and indirect lighting. The planar assumption additionally generates segmentation of non-planar objects and foreground texture as a by-product of the decomposition, which can potentially provide useful information for other tasks.

3 Problem and Assumptions

The input to our system is an RGB-D sequence capturing the indoor environment and a reconstructed mesh using the volumetric fusion method [24], [25]. The model is processed in three steps: (1) finding the geometric representation with planar abstraction (Section 4), (2) estimating texture (Section 5), and (3) setting the light parameters (Section 6).

We first detect planes on the reconstructed mesh based on the assumption that the indoor environment is composed of planar structures. It is a widely used assumption, and its success has been demonstrated in various indoor-based applications. Then, we further extend the assumption and extract the bare-bone structure of the indoor environment that encloses the given geometry. Based on the extracted structure of the room, the planes are classified as the room geometry or other structures. The extracted planes of the room geometry are extended, or connected. On the other hand, the remaining non-room geometry are treated separately. These are possible candidates for non-static objects with semantic meaning.

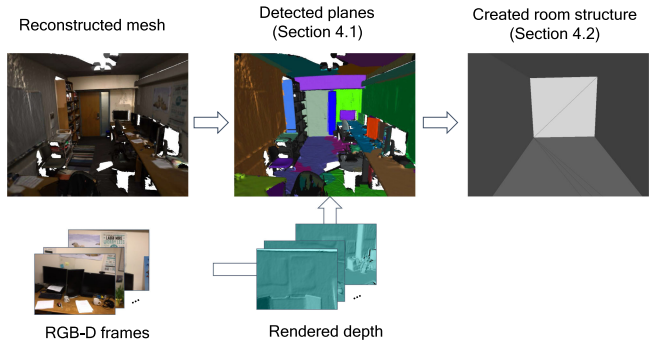


Fig. 2. The overview of geometry estimation.

Second, we cluster the colors of each plane and assign a background color with dominant homogeneous reflectance under a Lambertian assumption. The reflectance of an individual plane is estimated by the homogeneous background region, which is the region with no response under a conventional edge detector. We first compensate for color differences caused by per-frame auto-exposure using geometric correspondences between the mesh and the frames. Then the equalized color values are used to create texture on the mesh. From the mesh, we can locate the positions of direct lights and assign background reflectance colors for each plane. The per-plane texture is completed by optimizing the pose in the foreground and by filling the unobserved texture with the background color. At the same time, we estimate direct and indirect light parameters using the observed parts with the homogeneous texture.

The final output is the 3D content of the indoor environment, which is greatly simplified from the original sensor sequence or initial mesh. The lightweight mesh is complete and visually more attractive, with a clear texture and filled holes. We also have a geometric context of the 3D elements distinguishing the room structure of the essential static elements of walls, a ceiling, and a floor with representative background reflectance and other objects or foreground decorations. Also, we obtain the indirect light field as well as direct lighting. In short, the pipeline produces the necessary information on geometry, texture, and lighting utilizing the planar assumption, which can be readily rendered or used for a variety of indoor visualization, navigation, and mixed-reality applications.

4 Geometry Estimation

The use of planar abstraction for indoor geometry has been reported in many previous publications. In this section, we briefly discuss how we incorporate existing techniques using Manhattan world priors. The overall pipeline is depicted in Fig. 2. We first detect planes in individual frames and label

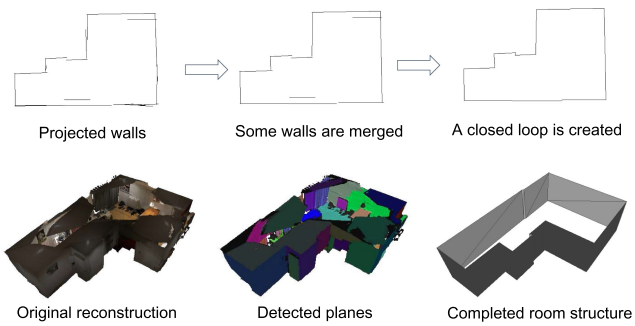


Fig. 3. Room structure estimation.

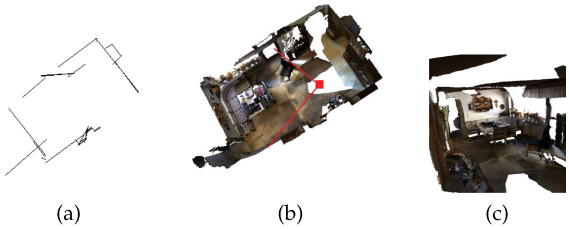


Fig. 4. The room completion step in Section 4.2 fails if there are holes near corners and the connectivity cannot be inferred easily. The projection of detected walls (a) cannot form a complete loop. From (c) [the view is illustrated in red in (b)], we can observe that the walls are lost as there are large windows and pathways.

the planes on the initial geometry (Section 4.1). We use this information to create a globally connected room structure and fill-in the missing geometry (Section 4.2).

4.1 Plane Detection and Refinement

The input to the system is a sequence of color-depth frames and their calibration information. We also have a 3D mesh built by volumetric reconstruction of the registered depth measurements. Instead of using the original depth measurements, which tend to be noisy, we generate the depth frames by reading the OpenGL depth buffer when the reconstructed mesh is rendered with the calibration parameters. From the rendered depth frames, we adapt the fast plane detection method suggested by [34]. By detecting the planes in individual frames instead of the 3D mesh, our method become robust against the large-scale distortion of planes that occur with the 3D mesh and effectively incorporate visibility information. The detected planes are then projected back to the reconstructed mesh, and the planes with sufficient overlap are merged into the same plane. The criteria for overlapping can be heuristic; however, we chose the case that if more than 10 percent of the detected plane pixels overlaps when projected on any of the frames. Examples of detected planes are shown in Figs. 2 and 3.

The plane parameters are found by running optimization on the detected planes, as mentioned by [16]. We find the plane parameters that are jointly optimized for the data fitting term and orthogonality. Further details are mentioned in Section 4.3 of [16].

In addition to the plane parameters, we also record the visibility information of individual planes. We project measurements of the individual frames on the plane and record the visibility information (existing, free-space, or occluded) considering the distance from the plane. This information is going to be used in the next section to complete the missing regions.

4.2 Plane Completion

Merely placing the detected planes is not enough to infer a complete room structure. While previous works detect corners and edges to generate a compact planar representation for visualization [12], [16], we found that the choices of connection between neighboring planes are usually ad hoc, and often result in wrong configurations. The main reason is that the observed data is distorted and occluded, missing the necessary connectivity information. The cause of the incompleteness is complex and merging or connecting planes with a constant distance threshold results in an erroneous decision. The connection is in contrast to the plane merging in Section 4.1, where the obviously overlapping planes are marked as the same. Open spaces of doorways, windows, or pillars make the problem even more complicated.

Given the high-resolution mesh, we need to enforce a strong prior to ultimately synthesize a simplified structure. At a high-level, our approach is similar to urban scene modeling [35], but the essential prior being used is different. For urban buildings, they utilize structural clues from GIS footprints, satellite imagery, and semantic segmentation and infer combinations of sweep-edges. For the indoor environment, we rely solely on RGB-D measurements and volumetric reconstruction, and we define individual rooms as space enclosed by a ceiling, a floor, and a loop of walls and enforce a stronger prior to generate the room structure. Currently, we focus on the single dominant loop, but we can extend the algorithm to sequentially add additional loops if needed. The decisions of extrapolating and merging planes are bold in the beginning but become more conservative in the later stages.

Sequentially, we first detect the ceiling and floor, followed by the largest enclosing loop of the walls from the detected planes and the remaining planes are added last. Specifically, we first detect ceiling and floor using the known gravity direction. Then we define candidate walls to be vertical planes that can extend from the ceiling to floor considering visibility information. The candidate walls are projected on the floor plane as lines with two ends, as shown in Fig. 3 (top left). We first merge some of the proximal lines that can be represented as a single wall (Fig. 3, top middle). This is because large planes are often partially observed by the sensor and thus are measured as several disconnected patches. The end-points of the lines are connected to create a loop, to extrapolate or to add new lines as necessary (Fig. 3, top right). The 2D projection can then be converted into a room structure. The described pipeline finds the room structure in most cases, except when there are windows of significant size and hall-ways as in Fig. 4. Once the loop of the room is defined, the remaining planes are conservatively connected when the intersection is observed.

The ceilings and floors are converted into 3D mesh with the help of a polygon triangulation library,[1] and the surrounding vertical rectangular walls are represented with two triangles. Among the remaining planes, the ones that can be fitted into rectangles are also represented with two triangles. The other remaining planes are meshed using a quad-tree structure on the plane (Fig. 3, bottom right). The

1. https://github.com/mapbox/earcut

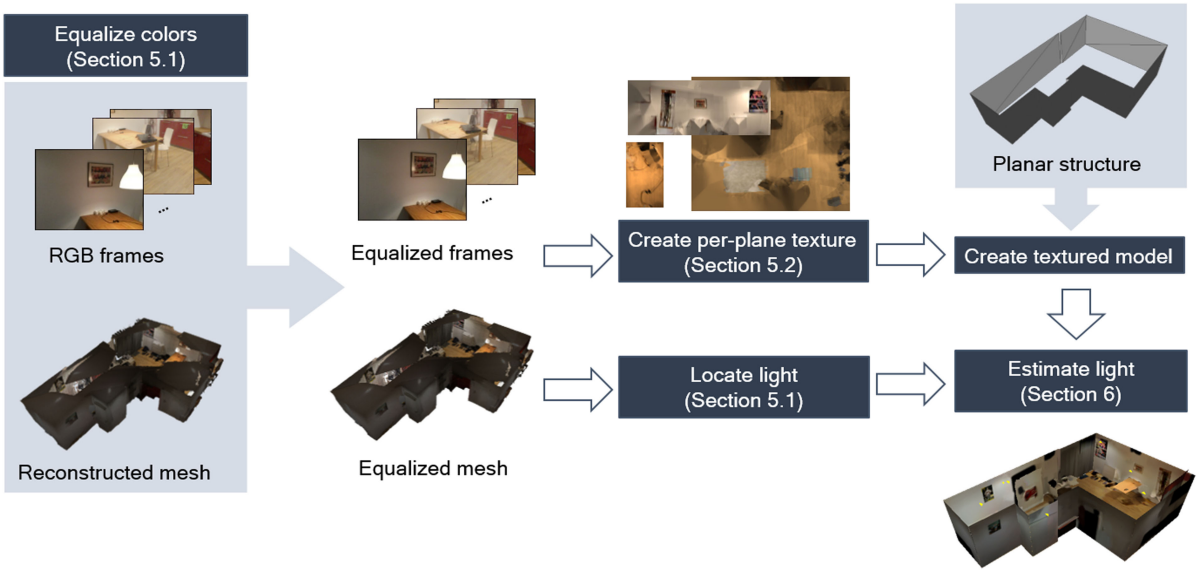


Fig. 5. Processing color and light.

overall size of the mesh is significantly reduced because 3D reconstruction from the volumetric fusion of sensors is usually based on a marching-cube algorithm [36] and creates triangles in the resolution of the voxel grid (Fig. 3, bottom left and bottom middle). In the next section, we create a complete high-resolution texture for the simple mesh to generate visually compelling 3D content.

5 Color Estimation

The previous section describes how we extract simplified geometry that captures the essential elements of the indoor environment. Having a rough geometry can also guide the inverse problem of decomposition, such as intrinsic imaging or inverse rendering. In this section, the planar abstraction is exploited to improve registration and create high-resolution texture. We first explain how we equalize colors by compensating for auto-exposure and white balancing. The equalized color information is used to detect light and to generate texture per plane by registering camera positions. The texture is then segmented into either a detailed foreground or uniform background color. The background color is used to fill the occluded region and solve for light parameters using simplified geometry. These stages are described in detail below, and also depicted in Fig. 5.

5.1 Color-Transfer Optimization

The texture can be generated by projecting multiple frames onto planes using the known calibration parameters. However, the corresponding pixels in different frames are not the same color, even on Lambertian surfaces, because of auto-exposure or white balancing. We compensate for the different exposures of the same geometric correspondence using the method suggested by [33]. We find corresponding pixels on images on sampled vertices $\mathbf{p}_k$ on the initial reconstructed mesh. We solve the following optimization problem:

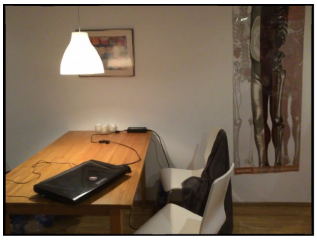
$$\min_{t_i, C(\mathbf{p}_k)} \sum_i (t_i C(\mathbf{p}_k) - I_i(\mathbf{p}_k))^2, \tag{1}$$

where t_i are per-image exposures, $C(\mathbf{p}_k)$ are vertex radiances, and $I_i(\mathbf{p}_k)$ is the gamma-corrected ($\gamma = 2.2$) observed pixel value of vertex $\mathbf{p}_k$ in image i. We only use correspondences that are not occluded and that are projected onto low-gradient regions in the images and solve for R, G, and B independently. Note that we did not use locally varying transfer functions as suggested by [16], [37]. This is because we are interested in emulating the actual color transfer by exposure or white balance with per-frame parameters to maintain the global relative brightness of the entire geometry. In practice, there exist non-Lambertian surfaces and outliers due to registration errors or geometric errors in reconstruction. Therefore, we ignore vertices $\mathbf{p}_k$ for which observation $I_i(\mathbf{p}_k)$ varies by more than 0.2 when the colors are converted to the range of $0-1$. We optimize the nonlinear least squares problem using the Huber-loss function with $\alpha = 0.1$. The details on implementation can be seen in the results section (Section 7).

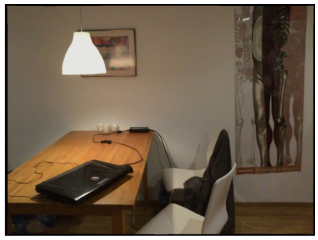
We assign the per-vertex reflectance using the correspondence calculated by projecting each vertex position onto individual frames. As the equalized colors cover a wider-range of color variation with the effect of different white-balance values of different frames, the resulting range of vertex colors involves high-dynamic-range radiance values. If there are multiple corresponding pixel values of the vertex, we assign confidence weights based on saturation and the geometric arrangement between the vertex and the camera, as suggested by [33]. The vertex color chosen is the weighted median of the equalized colors of corresponding pixels.

The high-dynamic range mesh can then be used to locate the direct, diffuse lights. To detect lights, we detect vertices whose radiances are above a threshold value, and identify the connected components. The center of mass of the connected mesh is chosen to be a direct light source. The calculated frame weights t_i and the locations of direct light sources are used in the later sections.

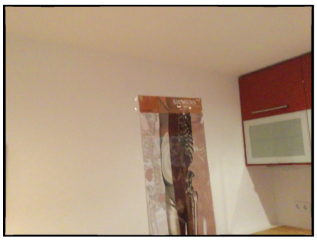
The sample results of color equalization are shown in Fig. 6. The input frames in Fig. 6a and 6c observe the same wall, but the white balance and exposure vary as the sensor



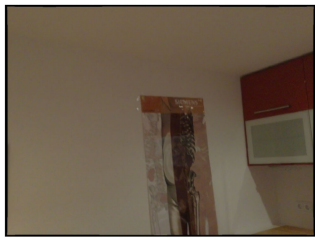
(a) Input frame 140



(b) Color-equalized frame 140



(c) Input frame 260



(d) Color-equalized frame 260

(e) Mesh texture with original input frames

(f) Mesh texture with equalized frames

Fig. 6. The results of the color-equalization step in Section 5.1.

automatically adjusts the exposure on a per-frame basis. Our color-equalization pipeline effectively reduces the color discrepancy, as shown in Fig. 6b and 6d. As a result, the mesh texture in Fig. 6f is more coherent than the original mesh texture in Fig. 6e. Also, the texture assigned with the median of observations greatly reduces blurry texture or ghosting artifacts due to misalignments.

5.2 Per-Plane Texture Generation

5.2.1 Per-Plane Registration

Similar to the mesh colors, the texture of the indoor structure for the simplified planes can be generated from the projection of RGB frames on the planes. With the known registration and 3D location of planes, this could be solved using simple homography. However, the initial geometry would be distorted, and the planar structure would be reconstructed as non-flat geometry. Consequently, the initial registration to the distorted geometry should be erroneous.

We solve for the camera-to-world 4-by-4 registration $\mathbf{T} = \{T_i\}$ of each frame. Individual frames in which the pixels correspond to a specific plane are further mapped to the plane with T^p, which transforms the world coordinate system into the plane coordinate system. In our setting, the transform T^p is defined such that one meter is mapped into 200 pixels. We use uniform scaling and rigid transformation to map the individual plane whose x and y coordinates represent the pixel coordinates and $z = 0$. We want to find a small update for the frame transformation $T_i = \Delta T_i \cdot T_i^0$, where $\mathbf{T}^0 = \{T_i^0\}$ represents the initial transformation. Assuming the update required for actual registration is small, we have

$$\Delta T_i \approx \begin{bmatrix} 1 & -\Delta\theta_z & \Delta\theta_y & \Delta t_x \\ \Delta\theta_z & 1 & -\Delta\theta_x & \Delta t_y \\ -\Delta\theta_y & \Delta\theta_x & 1 & \Delta t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Then for a point in a camera frame $\mathbf{p}^{camera}$, the point is mapped to a plane by $\mathbf{p}^{plane} = T^p \cdot T_i \cdot \mathbf{p}^{camera}$.

We first refine the registration of the individual frames by jointly solving for the planarity of geometry, sparse, and dense constraints. The formulation is similar to [16], but the individual terms are evaluated on the plane coordinate system:

$$E(\mathbf{T}) = E_g(\mathbf{T}) + \lambda_s E_s(\mathbf{T}) + \lambda_d E_d(\mathbf{T}). \tag{3}$$

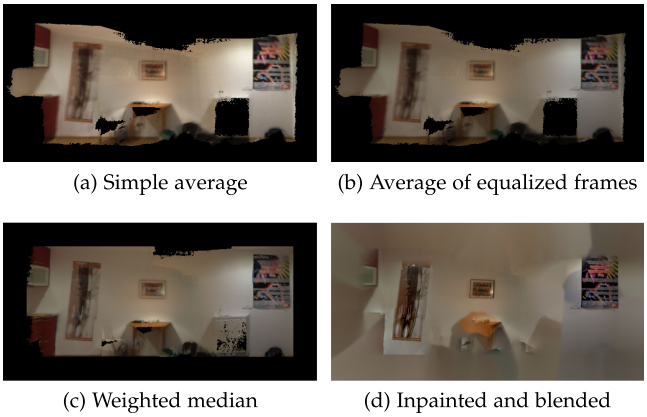
The weights λ_s and λ_d are set for each plane such that $E_g(\mathbf{T}^0) = \lambda_s E_s(\mathbf{T}^0) = \lambda_d E_d(\mathbf{T}^0)$.

The first term $E_g(\mathbf{T})$ represents the geometric term to hold the measurements close to the detected plane. When defined on the plane's coordinate system, we can minimize the distance to the plane by minimizing the z-coordinate. In other words, the geometric term can be written as:

$$E_g(\mathbf{T}) = \sum_i \sum_k \|\mathbf{p}_{i,k}^{plane} \cdot \mathbf{e}_3\|^2, \tag{4}$$

where $\mathbf{e}_3 = (0, 0, 1)$ represents the unit-vector in z-direction, and the term is minimized for all corresponding points (indexed by k) $\mathbf{p}_{i,k}^{plane}$ for projecting all frames i.

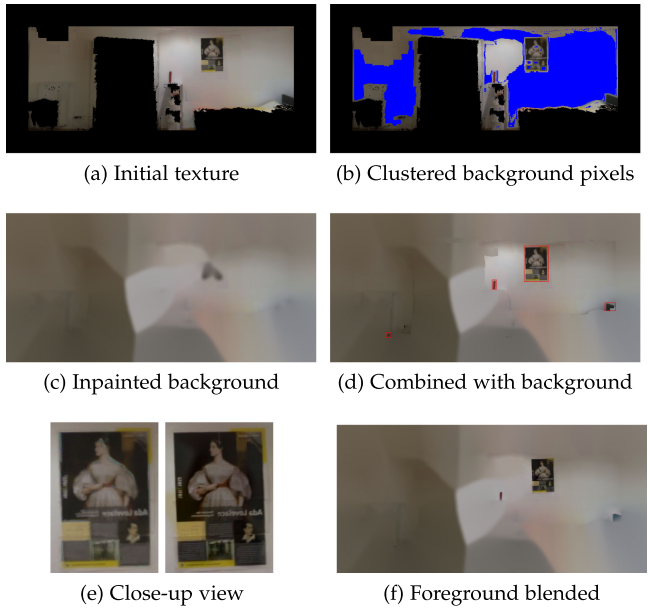
The second term $E_s(\mathbf{T})$ optimizes for the locations of sparse image feature correspondences. As before, all frame images are warped onto the plane, and sparse features are extracted and matched for every pair of (i, j) frames. We use the OpenCV implementation [38] of the ORB feature detector [39] and the brute-force matching strategy. We sort the matches by scores and keep only 1/3 of the found matches. We further filter out matches that are too far away when mapped onto the same plane (we use 10 pixels). We empirically found that the matches can be unstable if we have only one or two filtered matches. When there are more



(a) Simple average (b) Average of equalized frames

(c) Weighted median (d) Inpainted and blended

Fig. 7. Comparison of methods combining multiple frames to generate a texture.



(a) Initial texture (b) Clustered background pixels

(c) Inpainted background (d) Combined with background

(e) Close-up view (f) Foreground blended

Fig. 8. Results of the per-plane texture generation step in Section 5.2.2.

than three matches, we add the sparseness term to the optimization:

$$E_s(\mathbf{T}) = \sum_{i,j} \sum_{k} \|(\mathbf{p}_{i,k}^{plane} - \mathbf{p}_{j,k}^{plane}) \cdot (\mathbf{e}_1 + \mathbf{e}_2)\|^2. \quad (5)$$

We optimize the x, y coordinate of the location projected on the plane.

Furthermore, the third term $E_d(\mathbf{T})$ is optimized over the dense photometric consistency of individual pixel values $C(\mathbf{p}^{plane})$ in the generated texture. If $I_i(\mathbf{p}_{i,k}^{plane})$ represents the pixel intensity of a point in the color-corrected and warped image of frame i, it can be written as

$$E_d(\mathbf{T}) = \sum_{i} \sum_{k} \|C(\mathbf{p}_{i,k}^{plane}) - I_i(\mathbf{p}_{i,k}^{plane})\|^2. \quad (6)$$

For the dense term and geometric term, considering the points on the entire plane can be too expensive to compute without much constraint added as 3D coordinates on the detected plane are continuous. We chose to use uniformly sampled points to reduce the problem size. When optimized using a point for every five pixels in both x and y direction, it took about five minutes to run the optimization for each plane. The sampling rate can be adjusted considering the number of frames involved and the available memory size for the optimization.

5.2.2 Foreground-Background Optimization

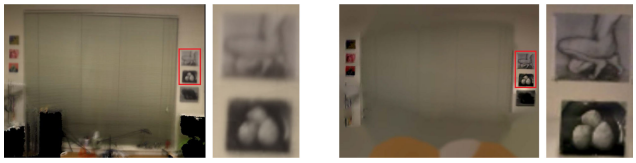
Color Blending. Even though we resort to simple geometry, we can still create the illusion of a realistic environment by rendering the model with a high-resolution texture. We choose the resolution of each texture based on the physical dimensions of individual planes. The color of each pixel is chosen in a manner similar to the generation of the mesh texture in Section 5.1, i.e., warping equalized colors of frames then taking the weighted median.

Fig. 7 shows a comparison of the different blending methods used to create texture from multiple frames. Compared to the simple averaging in Fig. 7a, using frame weights (Fig. 7b) compensates for different white balance values and consequently reduces the prominent boundaries of warped frames. Using a weighted median (Fig. 7c) eliminates additional ghosting and blurry details. However, there are still possible misalignments due to geometric errors or motion blur, and, more importantly, values missing from the geometry extrapolated which are not directly observed. The artifacts are reduced after we process the texture with inpainting and Poisson-blending as described in the following paragraphs (Fig. 7d).

Background Inpainting. We take a two-step approach for the background and the foreground. The underlying assumption is that each plane has a dominant color (base texture or background color) that can be smoothly interpolated and used to inpaint the missing regions. On the other hand, there are high-frequency details on top of the base texture, which are assigned as the foreground. The texture refinement steps for the foreground and background regions are described in Fig. 8.

An example of the combined initial texture is shown in Fig. 8a. We first cluster the RGB values of the pixels in the generated texture, avoiding the edge region. We collect dominant clusters that cover more than 60 percent of the non-edge pixels and mark them as background pixels. An example of background clusters is shown in Fig. 8b. We use the selected pixels to create the inpainted background (Fig. 8c). We used the OpenCV implementation of inpainitng [40] because our main focus was not on creating the best inpainting method, but a better approach can be used to fill larger regions. The inpainted background region is merged with the initial texture to create a full texture without missing values (Fig. 8d).

Foreground Blending. After the texture is filled with the background, the blurry foreground region is sharpened by blending images from the sharpest frame. As opposed to the background, the foreground regions have high-frequency details that need to be preserved. We first define the regions to be amended by finding connected components of detected edges. The samples of the regions are shown in Fig. 8d. For each connected component, we extract the best candidate frame to fill the part by selecting the warped texture that covers most of the region. If multiple frames cover the entire region, then we pick the frame that is the least blurry based on the variance of the Laplacian measure, as suggested by [41]. The region is then merged into the plane texture



(a) Average (b) Inpainted and blended

Fig. 9. Texture generation steps when the foreground/background assumption fails.

using Poisson blending [42] (Fig. 8f). As observed in Fig. 8e, the crisp details are successfully preserved.

The approach described fails when our underlying assumption does not hold. In particular, there are cases in which the background texture is not homogeneous, as depicted in Fig. 9. According to our foreground assumption, details are preserved for isolated pictures or decorations, as shown in the cropped region of Fig. 9b. However, the regular pattern of blinds is smoothed out as blurry gray fields. There are possible ways to improve the texture generation. The inpainting method can be replaced by traditional texture synthesis [43], [44], or even deep-learning-based approaches using GAN [45]. Our approach also cannot recover from the misalignment of non-flat global details. Huang *et al.* [16] suggest a careful non-rigid registration and graph-cut based method to create coherent registration and to create an assembly of patches. However, our approach with a homogeneous background assumption works with most flat walls in ordinary rooms of offices. More importantly, the background regions are represented by a simple reflectance model and can be used to estimate the necessary light parameters.

6 Light Parameter Estimation

In computer graphics, rendering is the process that generates images with geometry, light, and material. Inverse rendering [46], which solves the inverse problem, decomposes an image into light, material, and geometry. As in intrinsic image decomposition [30], [47] or illumination decomposition [48], [49], which are closely related problems in real-world images, inverse rendering is an under-constrained problem and is solved using a strong prior of one or more of the components, or by allowing user interactions.

In our set-up, we solve for light parameters under the assumption of planar geometry for a homogeneous background region. Starting from the low-level reconstruction of distorted mesh with RGB-D measurements, we extend the knowledge about planar geometry and can create full 3D content that ont only contains geometry, but also reflectance and light parameters representing both direct and indirect lighting. Geometric form factors of the radiance equation can be resolved using the simplified geometric estimation, resulting in a plausible solution for an under-constrained problem. With directional light sources, the light varies significantly for the different regions within the space, and our method can capture these effects. This is in contrast to light parameter estimation by previous methods only focused on small regions to place virtual objects, or considering only directions to represent an environment map. With fuller components that represent the environment, we can alter or insert any elements for rendering. We can seamlessly augment virtual objects, create realistic shadows, and change texture or lights.

6.1 Setting

Let us first begin with the famous radiance equation [50]:

$$R(V \rightarrow x') = \int_{x \in S} f(x \rightarrow V \rightarrow x') G(V, x) R(x \rightarrow V) dA. \tag{7}$$

The radiosity of a point V to another point x' is the sum of all radiance R received from other points x multiplied by BRDF f and visibility G. If we assume Lambertian reflectance, radiance is constant for any viewing direction, $R(V \rightarrow x) = R(V)$. This is approximated by the observed vertex radiance $C(\mathbf{p}_k)$ found in Section 5.1, and the BRDF can be written as a constant $f(V)$. We further assume that detected planes are represented with Lambertian homogeneous reflectance $f(V) = \rho$ in the background region, or in other words, where no high-frequency texture detail is observed. Then for a non-emitting vertex $\mathbf{p}_k$ the equation above can be re-written for a discretized mesh as

$$\begin{aligned} C(\mathbf{p}_k) &= \rho \sum_j G(\mathbf{p}_k, L_j) R(L_j \rightarrow \mathbf{p}_k) \\ &= \rho \left\{ x(\mathbf{p}_k) + \sum_j D(L_j \rightarrow \mathbf{p}_k) \right\}. \end{aligned} \tag{8}$$

Therefore, the pixel intensity $C(\mathbf{p}_k)$ is a combination of direct lighting $\sum_j D(L_j \rightarrow \mathbf{p}_k)$ and indirect lighting $x(\mathbf{p}_k)$. The models of direct lights and indirect lights are described below.

6.2 Direct Light

The locations of direct light are detected as bright regions after the vertex colors are equalized as described in Section 5.1. For each detected location, we place point-light sources whose light distribution is axially symmetric with the vertical axis of symmetry, where the vertical axis is defined as the normal of the ceiling and floor detection in Section 4.2. This is how most radially symmetric lights in the IES (Illumination Engineering Society) formats are used [51]. The profile can represent many types of lights found in residential indoor scenes, such as standing lights, ceiling lights, spotlights, and shaded lamps [33]. The intensity of light is dependent on the angle θ_{L_j} to the vertical axis, and the angular variation is represented by a 32-bin discretization of the angle θ_{L_j}.

The actual brightness at $\mathbf{p}_k$ depends on the reflectance of the wall, angle with respect to the normal of the wall $\theta_{\mathbf{p}_k}$, and the distance r to the point, but only when the path from the light source L_j to the point $\mathbf{p}_k$) is not occluded. Such occlusions are represented by a binary visibility function $G(\mathbf{p}_k, L_j)$.

To summarize, direct lighting can be written as:

$$D(L_j \rightarrow \mathbf{p}_k) = G(\mathbf{p}_k, L_j) \cdot I_{L_j} \cdot A_{L_j}(\theta_{L_j}) \cdot \cos\theta_{\mathbf{p}_k} \cdot \frac{1}{r^2}. \tag{9}$$

We only need to solve for the light intensity, which is formulated as the combination of the RGB factor I_{L_j} (3 dimensions) and angular 32-bins $A_{L_j}(\theta_{L_j})$. Other terms ($G(\mathbf{p}_k, L_j)$, θ_{L_j}, $\theta_{\mathbf{p}_k}$, and r) are geometric form factors and can be calculated using known information. For $G(\mathbf{p}_k, L_j)$, we locate the viewing camera at the light location, looking at individual planes and render the original input mesh. The depth buffer is used to detect visible points. For example, to find the light parameters for the hanging light in Fig. 10 (left), we can use the points on the table plane. The original mesh is first rendered with a virtual camera

Fig. 10. Precalculating the weights for light parameter estimation.

located at the light position, looking at the plane, as shown in Fig. 10 (middle). We can compute the visibility using the depth map and comparing the distances. The pre-calculated distance-based weights are depicted in Fig. 10 (right).

6.3 Indirect Light

The physically correct way to simulate indirect lighting is to run ray tracing multiple times with the correct geometry and material, until convergence. When converged, the field of indirect light can be represented by the position and the direction at any point within the volume. The full ray tracing involves a prohibitive amount of calculation and memory. Instead, indirect lights are often pre-calculated or approximated based on the fact that, unlike direct light, indirect light is smoother and subtler. The focus in previous literature has been on creating artificial light to effectively model the global illumination using, for example, ambient occlusion, an environmental map, or photon mapping, to name a few.

Modern VR applications usually use an environment map which indexex the normal directions to find the lighting on the face of a virtual object. Such approximation might work well when the environment is convex and confined, such as in a small rectangular room, but cannot work with more dynamic changes in the location. Instead, we assign unknown indirect lighting at each vertex $x(\mathbf{p}_k)$ and add a smoothness criterion for neighboring vertices. In other words, we minimize the following optimization problem:

$$\sum_k \left\| C(\mathbf{p}_k) - \rho \cdot \left\{ \sum_j D(L_j \rightarrow \mathbf{p}_k) + x(\mathbf{p}_k) \right\} \right\|^2 + \lambda_c \sum_{(k_1,k_2)\in N} \left\| x(\mathbf{p}_{k_1}) - x(\mathbf{p}_{k_2}) \right\|^2. \tag{10}$$

The first term matches the color at the vertex with the light and reflectance, and the second term enforces smoothness between neighboring indirect lighting. For implementation, we regularly sample vertices on the plane region clustered as the background.

6.4 Implementation

We solve Equation (10), and the lower and upper bounds of all unknowns are set to be between 0 and 1. Because the problem is under-constrained with many unknowns, we impose constraints on the initialization and solve alternative optimization by grouping variables. We initialize the light intensity using the absolute difference of shadows in one of the close boundaries, and is constant in all directions. The reflectance is initialized as the median of the background texture, and indirect lighting is set to $\{C(\mathbf{p}_k) - \sum_j D(L_j \rightarrow \mathbf{p}_k)\}/\rho$. We alternatively optimize for light intensity, reflectance, and indirect lighting, and use the closest three visible planes per light. The imposed assumptions are sometimes violated due to varying reflectance, simplification of indirect lighting, and incorrect geometry or texture. We, therefore, use the Huber function as a robust loss function.

TABLE 1
The characteristics of the original indoor data sets built with the VoxelHashing approach[53] whose representation has been reduced either by 3dlite [16] or our pipeline

		3dlite		BundleFusion	ScanNet
	scenes	apt	office0	office3	0567_01
initial	frames	2865	6159	3820	2066
	faces	3,291,072	926,414	3,045,096	435,126
3dlite	planes	32	20	18	13
	faces	92,898	62,688	63,479	43,153
	reduction	2.8%	6.7%	2.1%	9.9%
ours	planes	30	23	29	15
	faces	21,092	9,490	33,370	11,054
	reduction	0.64%	1.0%	1.1%	2.5%

While we cannot indicate the quality solely with the numbers, our algorithm produce a comparable number of planes with smaller number of faces.

TABLE 2
The run time (in seconds) of different stages of the pipeline

	3dlite	BundleFusion		ScanNet
scenes	apt	office0	office3	0567_01
plane detect.	3,925	5,409	5,362	6,513
plane compl.	0.528	122	27	11
color eq.	342	405	494	360
texture gen.	1,229	467	1,133	308
light est.	6,761	2,711	1,705	834
total	12,257 (3.4h)	9,114 (2.5h)	8,721 (2.4h)	8,026 (2.2h)

Each of the components represent plane detection (Section 4.1), plane completion (Section 4.2), color equalization (Section 5), texture generation (Section 5.2), and light estimation (Section 6).

7 Results

The pipeline has been implemented in a desktop machine with Intel Core i7 3.6GHz CPU with 16 GB memory. We used the Ceres solver [52] for variation optimization problems involved in the registration, color, and light

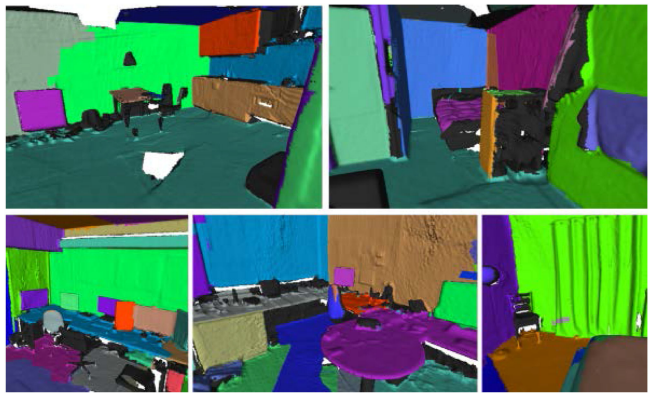


Fig. 11. Plane detection parses the complex geometry of indoor spaces and the remaining objects can further be classified into isolated clusters.

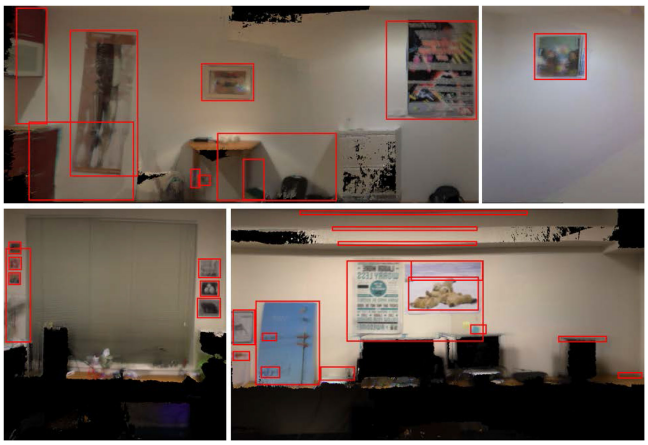


Fig. 12. The detected non-background region to be inpainted.

optimization. The rendering of virtual scenes is implemented using the unreal engine with point-light sources. We tested the pipeline with sequences available from [16], the BundleFusion [25], and the ScanNet [24] data set. The initial

Fig. 13. Provided the acquired geometry and light information, the original video is augmented with a virtual object using differential rendering. Note that the shadow resulted from the directional light above the virtual object.

mesh is built with the VoxelHashing approach [53]. The details of the dataset used are available in Table 1. After about 2-3 hours of processing, the complete and lightweight representation is acquired with only 1-6% of the face elements. Our algorithm produces a comparable number of planes with a smaller number of faces. The major reduction discrepancy is from the meshing approach, which can be seen by comparing the third and fifth row of Fig. 14 and 15.

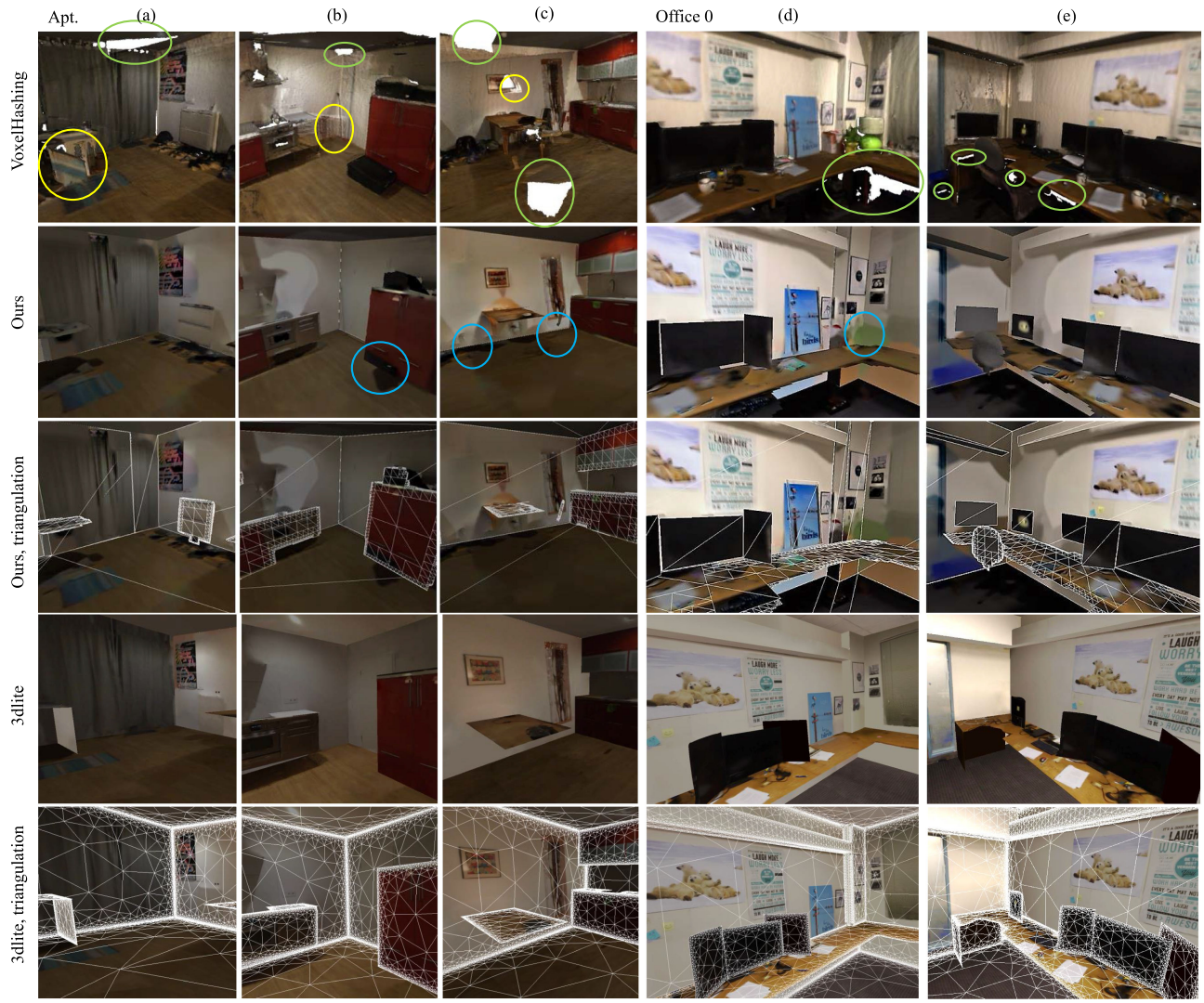


Fig. 14. Comparison of visualization using VoxelHashing [53] (top rows) with our representation (middle and bottom rows). The volumetric reconstruction suffers from ghosting (yellow) or holes (green), and our approach alleviates the artifacts. In the meanwhile, our approach erases non-planar objects (blue) and fill it with nearby background colors. Our representation uses much smaller number of triangles (bottom rows), but exhibit crisp texture for detected foreground.

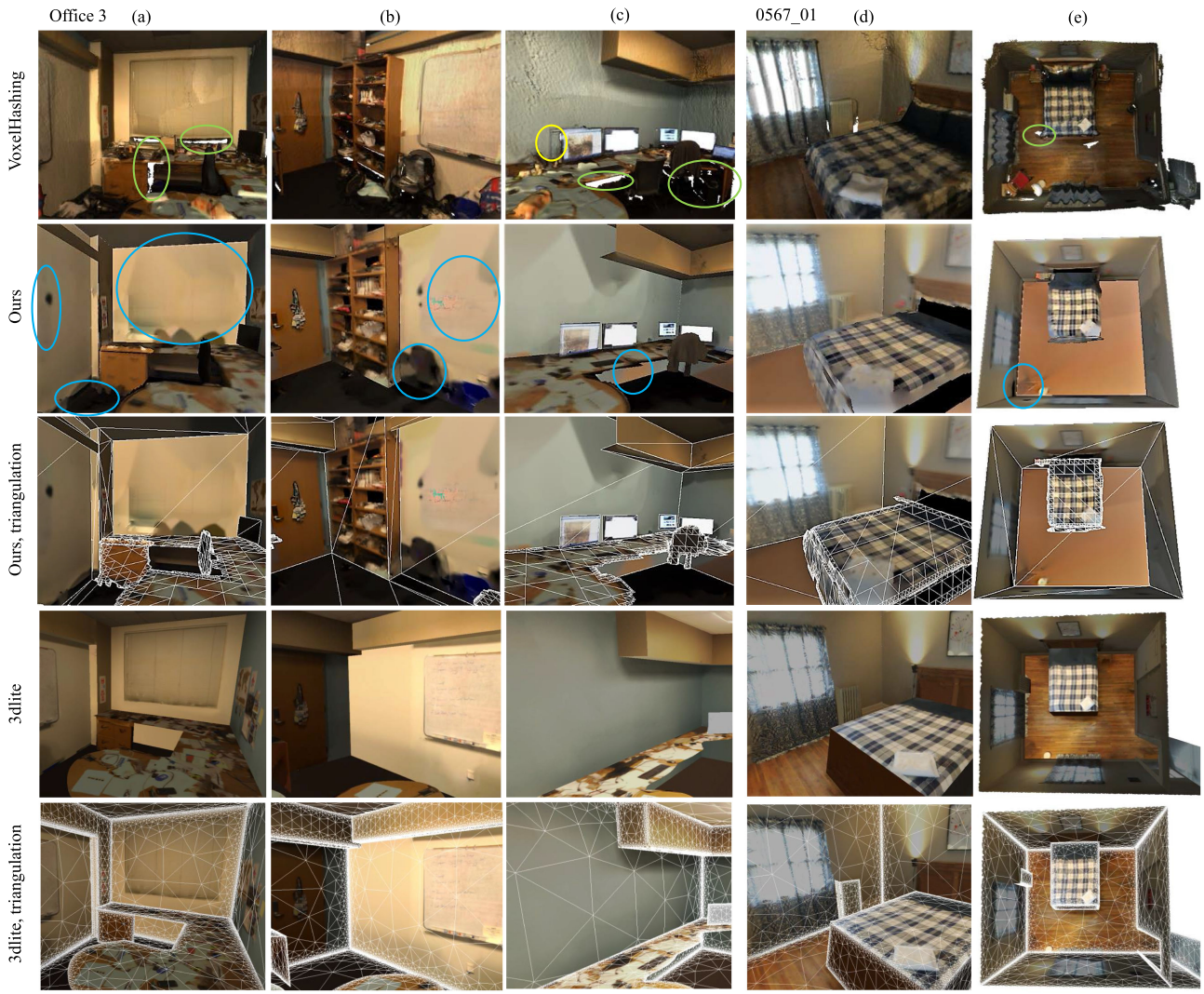


Fig. 15. Comparison of visualization using VoxelHashing [53](top rows) with our representation (middle and bottom rows). The volumetric reconstruction suffers from ghosting (yellow) or holes (green), and our approach alleviates the artifacts. In the meanwhile, our approach erases non-planar objects (blue) and fill it with nearby background colors. Our representation uses much smaller number of triangles (bottom rows), but exhibit crisp texture for detected foreground.

Details of the timing are available in Table 2. The plane detection and light estimation steps require the most computation.

In this section, we focus on the visualization of the generated 3D content. (While the process is composed of multiple stages, we described the effects and performance of individual components in the respective sections.) Figure Fig. 14 and 15 show the qualitative comparison between the original volumetric reconstruction (first row) and our lightweight reconstruction (second and third row). With a fraction of elements, we can still convey the overall shape of the environment. Samples of reduced triangle faces are highlighted in the second and third rows of Figs. 14 and 15. More importantly, our pipeline significantly reduces the ghosting artifacts near the depth boundaries (shown in yellow) and fill unnecessary holes (shown in green). Most of the details on the texture are crisp, and the color stays equalized regardless of the varying white-balancing per frame. Our planar representation erases non-planar objects from the original reconstruction and fills the unknown texture of the revealed planar region with the background color. Some examples are highlighted in blue in Figs. 14 and 15.

We also compared the geometry and texture of our reconstruction against the 3dlite [16], whose results are presented in fourth and fifth row of Figs. 14 and 15. The reconstruction provided by the authors was in a slightly different coordinate system and we roughly aligned the viewpoint with our reconstruction. Although 3dlite is similarly focused on dominant planes, the approach is slightly different. The most prominent difference is that the choice of planes; for example, the white drawer in Fig. 14(a) and the bookshelf in Fig. 15(b) are detected only in our case. However, the side-board of the table in Fig. 14(a) and cupboard in Fig. 15(e) are only detected with 3dlite. The choice of retained planes becomes different largely because our algorithm emphasizes preserving the room structure, which is obviously presented observing the shape of the room for Fig. 15(e). On the other hand, 3dlite puts more emphasis on finding intersections of detected planes and has a smaller number of floating structure such as monitors or chairs (Figs. 14(e), 15(a)(c)). In terms of texture, 3dlite preserves sharper texture when the underlying geometry is not exactly plane, as they allow non-rigid warping of image

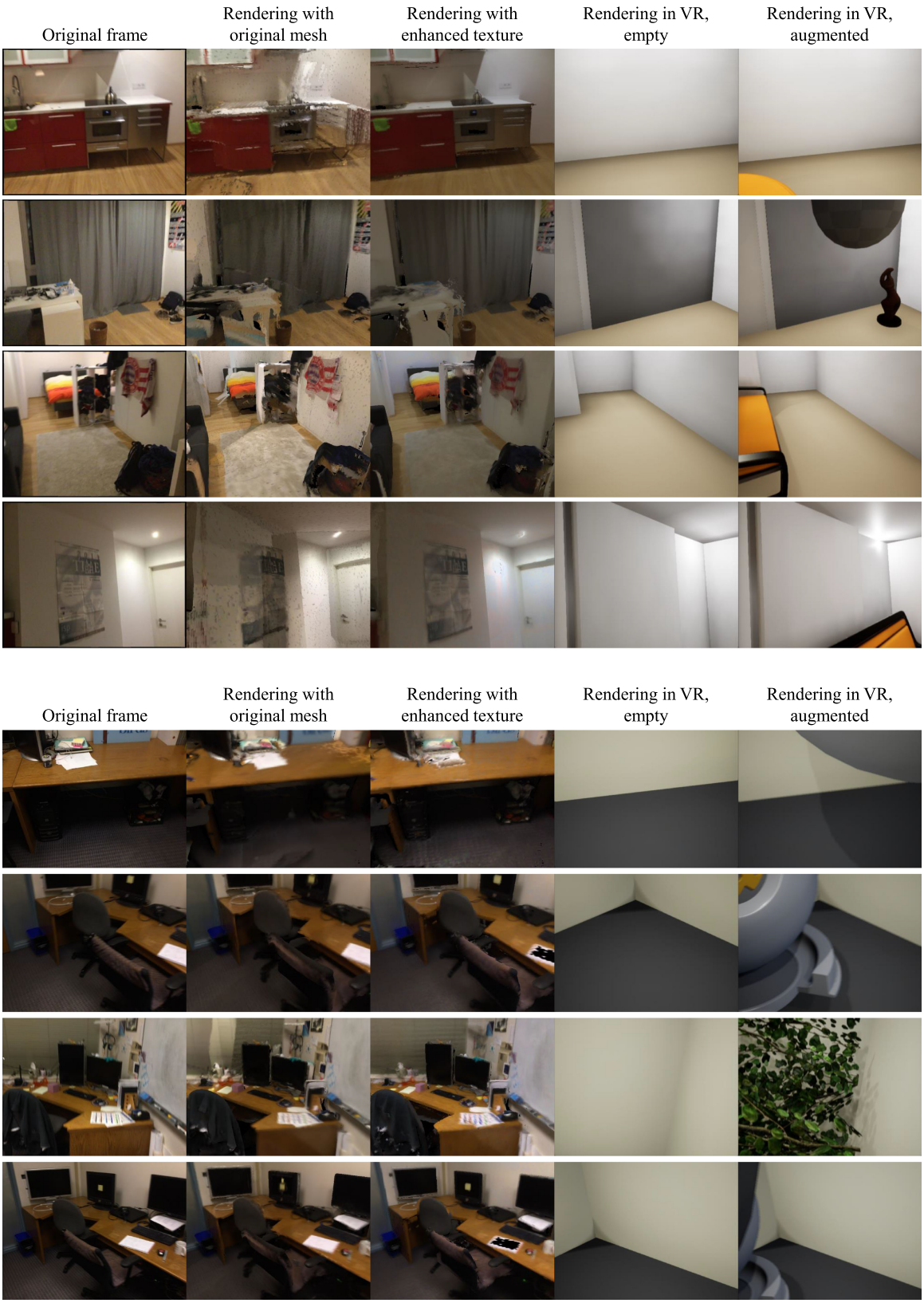


Fig. 16. Samples of input frames and the rendering at the same view for *apt* and *office0* data set.

patches during their texture generation. On the other hand, they do not correctly compensate for the light information and there are prominent differences in white-balance for different walls (Fig. 14(c)(e) and 15(b)). We focus on having the correct balance of reflectance and later use the color information to solve for the light information, which is not in consideration in 3dlite. The light information, in conjunction with the planar abstraction, can be used to augment the original video as in Fig. 13. Our HDR mesh combines multiple frames with a different original exposure into a single,

coherent texture with high-dynamic range, and makes it possible to locate light sources. As the result of the light estimation, the effect of directional light is visible with the shading and shadow of the augmented object in the AR example (more frames are available in the accompanied video).

With the retrieved planar proxies and light information, we have a simplified representation and the viewpoints can be quickly rendered. We compare the original input frames (first column) with renderings of the textured mesh in Fig. 16. The simple rendering of the transformed initial mesh reveals an imprecise texture due to the errors in the reconstruction and the registration (second column). The retexturing method using the weighted median of the equalized frames can alleviate such artifacts (third column). We then find the background colors of the planes and use the light location to solve the simplified inverse rendering problem. From the inferred values and using the simplified planar geometry, we can create a virtual scene of an empty room, that contains the detected loop of the walls with background colors (fourth column). The light variations near the light sources are captured with the color variation of the planes. The virtual scene can be freely altered for VR applications (fifth column).

One interesting bi-product of the pipeline is the geometric decomposition of various elements. The geometry estimation creates the decomposition of planes as depicted in Fig. 11. Most of the time, the decomposed planar elements agree with semantic segments, such as individual pieces of furniture, walls, ceilings, or floors. Nonplanar objects can also be segmented after planar objects are distinguished. Color estimation also enables foreground-background segmentation. The texture completion pipeline separates connected components for the foreground, which can be used to approximately locate different elements on the planar surface (Fig. 12).

There are a few exciting future directions. One is to incorporate the acquired room structure and the information of the non-planar geometry to assist intelligent agents in navigating and paying attention to detecting the unknown components [54], [55], [56]. Also, we could incorporate a physical understanding of the scene to complete the unseen parts of the environment [57]. Currently, some of the structure that is partially planar can remain floating in the air after our pipeline.

7.1 Limitations

The described pipeline works as post-processing of an initial 3D reconstruction captured by an RGB-D camera, which can also be a challenge for ordinary users. One can extend the work to apply as an on-line process with possible user interaction to quickly acquire the light-weight 3D content using planar abstraction [58].

The major limitation of the work comes from the major contribution of the work, which is relying on a very strong prior. The prior, while effective in many cases, limits the range of environments that can be reconstructed in terms of geometry, texture, and lighting. We can try representing more general scenes by relaxing the restrictions on each of the three components.

In terms of geometry, our suggested pipeline currently uses a single Manhattan-world frame, and an enclosed room composed of a single ceiling and a floor. The assumption excludes places such as large houses with multiple floors, or open spaces such as auditoriums or stadiums. We can extend the pipeline to cover general planes or other proxies, such as combining with CAD models and similarly fill the inevitable holes or artifacts.

The texture and lighting prior can be relaxed by applying techniques from intrinsic imaging. For example, we can apply texture segmentation and solve for texture, lighting, and shading. Currently, we only allow a simple point light assumption, but we can use more complex light models, including windows [33]. While the individual choices of the indoor scene prior can differ, we believe that jointly reasoning about geometry, texture, and light information is possible with a simplified representation of real-world geometry and can allow for light-weight, yet usable capture of the real spaces.

8 Conclusion

We presented a holistic pipeline to represent a captured indoor environment into 3D content ready for AR/VR applications with full geometry, texture, and lighting information. We first focused on completing the room structure based on plane detection. Individual planes were further refined, detecting dominant colors for the background. The detected backgrounds are used to fill in unobserved regions and extract the reflectance and the shading cues for inverse rendering, while the remaining foreground is used to refine the registration and create a crisp texture of the recovered geometry. The generated representation can be used to visualize and navigate the captured environment. From our knowledge, we are the first to suggest an approach to jointly consider geometry, texture, and light information for an indoor environment and convert a real 3D indoor environment into a complete virtual 3D asset.

Acknowledgments

This work was supported in part by the New Faculty Startup Fund from SeoulNational University, the ICT R&D program of MSIP/IITP [2017-0-00162, Development of Human-care Robot Technology for Aging Society], and the KIST institutional program [Project No. 2E29450].

References

[1] G. Optale *et al.*, "Controlling memory impairment in elderly adults using virtual reality memory training: A randomized controlled pilot study," *Neurorehabilitation Neural Repair*, vol. 24, no. 4, pp. 348–357, 2010. [Online]. Available: https://doi.org/10.1177/1545968309353328

[2] G. Riva, "Virtual reality as assessment tool in psychology," vol. 44, pp. 71–9, 1997.

[3] A. Rizzo *et al.*, "Virtual environment applications in clinical neuropsychology," in *Proc. IEEE Virtual Reality 2000 (Cat. No.00CB37048)*, 2000, pp. 63–70.

[4] Y. Zhang *et al.*, "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5057–5065.

[5] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, "MINOS: Multimodal indoor simulator for navigation in complex environments," 2017, *arXiv: 1712.03931*.

[6] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Found. Trends Comput. Graphics Vis.*, vol. 9, no. 1–2, pp. 1–148, Jun. 2015. [Online]. Available: http://dx.doi.org/10.1561/0600000052

[7] E. Prados and O. Faugeras, *Shape From Shading*. Boston, MA, USA: Springer, 2006, pp. 375–388. [Online]. Available: https://doi.org/10.1007/0-387-28831-7_23

[8] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017.

[9] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graphics Interactive Tech.*, 1996, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/237170.237269

[10] R. A. Newcombe *et al.*, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[11] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," , vol. 37, no. 6, Nov. 2018, Art. no. 257. [Online]. Available: http://www-sop.inria.fr/reves/Basilic/2018/HPPFDB18

[12] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra, "Rapter: Rebuilding man-made scenes with regular arrangements of planes," *ACM Trans. Graphics*, vol. 34, no. 4, pp. 103:1–103:12, Jul. 2015.

[13] Y. Zhang, W. Xu, Y. Tong, and K. Zhou, "Online structure analysis for real-time indoor scene reconstruction," *ACM Trans. Graphics*, vol. 34, no. 5, pp. 159:1–159:13, Nov. 2015. [Online]. Available: http://doi.acm.org/10.1145/2768821

[14] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 80–87.

[15] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," *Int. J. Comput. Vision*, vol. 110, no. 3, pp. 243–258, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s11263-014-0711-y

[16] J. Huang, A. Dai, L. Guibas, and M. Nießner, "3DLite: Towards commodity 3D scanning for content creation," *ACM Trans. Graphics*, vol. 36, 2017, Art. no. 203.

[17] Z. Liao, K. Karsch, and D. Forsyth, "An approximate shading model for object relighting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5307–5314.

[18] T. Rhee, L. Petikam, B. Allen, and A. Chalmers, "Mr360: Mixed reality rendering for 360 panoramic videos," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 4, pp. 1379–1388, Apr. 2017.

[19] K. Karsch *et al.*, "Automatic scene inference for 3d object compositing," *ACM Trans. Graphics*, vol. 33, no. 3, pp. 32:1–32:15, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602146

[20] "Arcore - get light estimation," [Online]. Available: https://developers.google.com/ar/reference/unreal/arcore/blueprint/Get_Light_Estimation. Accessed on: Aug. 3, 2018.

[21] P. Debevec, "Image-based lighting," *ACM SIGGRAPH 2005 Courses*, 2005. [Online]. Available: http://doi.acm.org/10.1145/1198555.1198709

[22] S. Jiddi, P. Robert, and E. Marchand, "Reflectance and illumination estimation for realistic augmentations of real scenes," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2016, pp. 244–249.

[23] M.-A. Gardner *et al.*, "Learning to predict indoor illumination from a single image," *ACM Trans. Graphics*, vol. 36, no. 6, pp. 176:1–176:14, Nov. 2017. [Online]. Available: http://doi.acm.org/10.1145/3130800.3130891

[24] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. Conf. Comput. Vis. Pattern Recogni.*, 2017, pp. 2432–2443.

[25] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Trans. Graphics*, vol. 36, 2017, Art. no. 76a.

[26] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," 2017, *arXiv:1702.01105*.

[27] C. Lipski, F. Klose, and M. Magnor, "Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 942–951, Jun. 2014.

[28] Y. Yu, P. Debevec, J. Malik, and T. Hawkins, "Inverse global illumination: Recovering reflectance models of real scenes from photographs," in *Proc. 26th Annu. Conf. Comput. Graphics Interactive Tech.*, 1999, pp. 215–224. [Online]. Available: http://dx.doi.org/10.1145/311535.311559

[29] B. Alexandre, M. de La Gorce, and M. Renaud, "Piecewiseplanar 3D reconstruction with edge and corner regularization," *Comput. Graphics Forum*, vol. 33, no. 5, pp. 55–64, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12431

[30] J. T. Barron and J. Malik, "Intrinsic scene properties from a single RGB-D image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 17–24.

[31] A. Teichman, S. Miller, and S. Thrun, "Unsupervised intrinsic calibration of depth sensors via SLAM," in *Proc. Robot. Sci. Syst.*, Jun. 2013, doi: 10.15607/RSS.2013.IX.027.

[32] J. Jachnik, R. A. Newcombe, and A. J. Davison, "Real-time surface light-field capture for augmentation of planar specular surfaces," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2012, pp. 91–97.

[33] E. Zhang, M. F. Cohen, and B. Curless, "Emptying, refurnishing, and relighting indoor spaces," *ACM Trans. Graphics*, vol. 35, no. 6, 2016, Art. no. 174.

[34] C. Feng, Y. Taguchi, and V. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6218–6225.

[35] T. Kelly, J. Femiani, P. Wonka, and N. J. Mitra, "BigSUR: Large-scale structured urban reconstruction," *ACM Trans. Graphics*, vol. 36, no. 6, 2017, Art. no. 204. [Online]. Available: https://doi.org/10.1145/3130800.3130823

[36] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proc. 14th Annu. Conf. Comput. Graphics Interactive Tech.*, 1987, pp. 163–169.

[37] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 70:1–70:9, 2011.

[38] G. Bradski, "The openCV library," *Dr. Dobb's J. Softw. Tools*, 2000.

[39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[40] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graphics Tools*, vol. 9, no. 1, pp. 23–34, 2004. [Online]. Available: https://doi.org/10.1080/10867651.2004.10487596

[41] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martínez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: A comparative study," in *Proc. Int. Conf. Pattern Recognit.*, 2000, vol. 3, pp. 314–317.

[42] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Proc. ACM SIGGRAPH 2003 Papers*, 2003, pp. 313–318. [Online]. Available: http://doi.acm.org/10.1145/1201775.882269

[43] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1999, pp. 1033–1038.

[44] Y. Liu, W.-C. Lin, and J. Hays, "Near regular texture analysis and manipulation," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 368–376, Aug. 2004.

[45] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang, "Non-stationary texture synthesis by adversarial expansion," *ACM Trans. Graphics*, vol. 37, no. 4, pp. 49:1–49:13, 2018.

[46] R. Ramamoorthi and P. Hanrahan, "A signal-processing framework for inverse rendering," in *Proc. 28th Annu. Conf. Comput. Graphics Interactive Tech.*, 2001, pp. 117–128. [Online]. Available: http://doi.acm.org/10.1145/383259.383271

[47] H. G. Barrow and J. M. Tenenbaum, "Recovering intrinsic scene characteristics from images," AI Center, SRI Int., Menlo Park, CA 94025, Tech. Rep. 157, Apr. 1978.

[48] R. Carroll, R. Ramamoorthi, and M. Agrawala, "Illumination decomposition for material recoloring with consistent interreflections," *ACM SIGGRAPH 2011 papers*, Aug. 2011, Art. no. 43. [Online]. Available: http://graphics.berkeley.edu/papers/Carroll-IDM-2011-08/

[49] S. M. Seitz, Y. Matsushita, and K. N. Kutulakos, "A theory of inverse light transport," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, 2005, pp. 1440–1447.

[50] M. F. Cohen, J. Wallace, and P. Hanrahan, *Radiosity and Realistic Image Synthesis*. San Diego, CA, USA: Academic Press Professional, 1993.

[51] I. C. Committee, *IESNA Standard File Format for Electronic Transfer of Photometric Data*. New York, NY, USA: Illuminating Eng. Society North America, 1995. [Online]. Available: https://books.google.co.kr/books?id=Gm_yAAAAMAAJ

[52] S. Agarwal, K. Mierle, and Others, "Ceres solver." [Online]. Available: http://ceres-solver.org

[53] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, 2013.
[54] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn., ser. Proc. Mach. Learn. Res.*, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2048–2057. [Online]. Available: http://proceedings.mlr.press/v37/xuc15.html
[55] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Adv. Neural Info. Process. Syst.* 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3545–3553.
[56] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu, "Recurrent models of visual attention," in *Adv. Neural Inform. Process. Syst. 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc.,2014, pp. 2204–2212. [Online]. Available: http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf
[57] T. Shao, A. Monszpart, Y. Zheng, B. Koo, W. Xu, K. Zhou, and N. J. Mitra, "Imagining the unseen: Stability-based cuboid arrangements for scene understanding," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 209:1–209:11, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661229.2661288
[58] Y. M. Kim, J. Dolson, M. Sokolsky, V. Koltun, and S. Thrun, "Interactive acquisition of residential floor plans," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3055–3062.

Young Min Kim received the BS degree in electrical engineering from Seoul National University, Korea, in 2006, and the MS and PhD degrees in electrical engineering from Stanford University, Stanford, CA, in 2008 and 2013, respectively. She is currently an assistant professor at Seoul National University (SNU), South Korea. Prior to that, she was a senior research scientist at the Center for Imaging Media Research, Korea Institute of Science and Technology (KIST), Seoul, South Korea. Her research interests include computer vision and computer graphics, especially in the area involving 3D information and practical applications.

Sangwoo Ryu is currently working toward an undergraduate degree at the Pohang University of Science and Technology (POSTECH). His research interests include computer graphics and computer vision.

Ig-Jae Kim received the BS and MS degrees from the Electrical Electronic Engineering of Yonsei University, Seoul, South Korea, in 1998 and 1996, respectively, and the PhD degree from the EECS of Seoul National University, in 2009. He is currently a director of Center for Imaging Media Research, Korea Institute of Science and Technology (KIST), Seoul, South Korea. He is also an associate professor at the Korea University of Science and Technology. He had worked in Massachusetts Institute of Technology (MIT) Media Lab as a postdoctoral researcher (2009-2010). He has published over 80 fully-referred papers in international journal and conferences, including ACM Transaction on Graphics, Pattern Recognition, SIGGRAPH, Eurographics, etc. His research interests include pattern recognition, computer vision and graphics, deep learning, and computational photography.