Planar Abstraction and Inverse Rendering of 3D Indoor Environment

Young Min Kim[1,2] Sangwoo Ryu[3] and Ig-Jae Kim[1]

[1]Korea Institute of Science and Technology (KIST), Korea
[2]Seoul National University, Korea [3]POSTECH, Korea

Abstract

A large-scale scanned 3D environment suffers from complex occlusions and misalignment errors. The reconstruction contains holes in geometry and ghosting in texture. These are easily noticed and cannot be used in visually compelling VR content without further processing. On the other hand, the well-known Manhattan World priors successfully recreate relatively simple or clean structures. In this paper, we would like to push the limit of planar representation in indoor environments. 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 produced content can effectively represent the spatial arrangements for various AR/VR applications and can be readily combined with virtual objects possessing plausible lighting and texture.

CCS Concepts

•***Computing methodologies*** $\rightarrow$ *Texturing; Mixed / augmented reality; Reflectance modeling;*

1. Problem and Assumptions

The input to our system is a RGB-D sequence capturing the indoor environment and a reconstructed mesh using the volumetric fusion method [DNZ*17, DCS*17]. The model is processed in three steps: (1) finding the geometric representation with planar abstraction (Section 2), (2) estimating texture (Section 3), and (3) setting the light parameters (Section 4). 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 visually more attractive with clear texture and filled holes. In addition, we obtain the indirect light field as well as direct lighting. In short, the pipeline produces the necessary information on geometry, texture, and lighting with the help of the planar assumption, which can be readily rendered or utilized for a variety of indoor visualization, navigation, and mixed-reality applications.

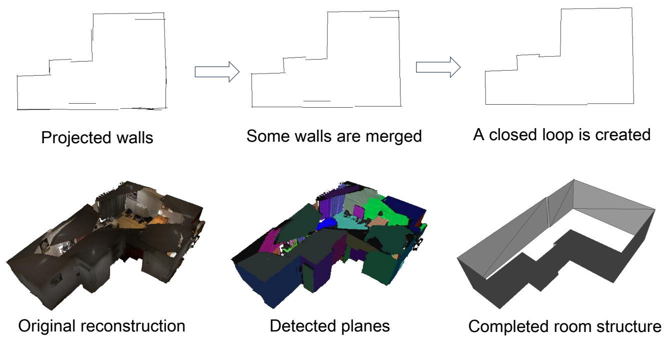

Figure 1: Room structure estimation.

2. Geometry estimation

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 tends 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 [FTK14]. By detecting the planes in individual frames instead of 3D mesh, we not only become robust against the large-scale distortion of planes that can occur with the long-term accumulation but also effectively incorporate the 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 (Figure 1).

The initial detected planes do not form a coherent structure. While previous works detect corners and edges to generate compact planar representation for visualization [HDGN17, MMBM15], we found that the choices of connection between neighboring planes are usually ad hoc, and often result in wrong configurations. The

DOI: 10.2312/egs.20191020

main reason is that the observed data is distorted and occluded, missing necessary connectivity information.

We sequentially first detect ceiling and floor, followed by the loop of the walls to define the room structure. The remaining walls 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 (https://github.com/mapbox/earcut), 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 quad-tree structure on the plane (Figure 1, bottom right). The overall size of the mesh is greatly reduced because 3D reconstruction from volumetric fusion of sensors is usually based on a marching-cube algorithm [LC87], which creates the triangles within the resolution of the voxels (Figure 1, bottom left and bottom middle).

3. Color Estimation

3.1. Color-transfer optimization

The planar abstraction extracted in the previous section is further exploited to improve registration and create high-resolution texture. The texture can be generated by combining the projection of multiple frames onto a plane 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 using geometric correspondence and the method suggested by [ZCC16]. 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 is per-image exposure, $C(\mathbf{p}_k)$ is the vertex radiance, and $I_i(\mathbf{p}_k)$ is the gamma-corrected ($\gamma = 2.2$) pixel value of a vertex $\mathbf{p}_k$ in the image i. The equalized high-dynamic-range mesh is created with the vertex color to be the weighted median of the equalized colors of the corresponding pixels.

The high-dynamic range mesh can then be used to locate the direct diffuse lights. To detect lights, we threshold by the vertex radiance, and detect the connected components. The center of mass of the connected mesh are chosen to be a direct light source.

3.2. Per-plane texture generation

3.2.1. Per-plane registration

Similar to the mesh colors, the texture of the indoor structure for the simplified planes can be generated using 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 clearly distorted from a perfect plane and, as a consequence, the initial registration to the distorted geometry should have been erroneous.

We solve for the local optimum of the camera-to-world registration $\mathbf{T} = \{T_i\}$ of each frame given the initial registration $\mathbf{T}^0$. 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 xy plane with the normal in the z direction. 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, in addition to the sparse, and dense constraints on the plane coordinate system:

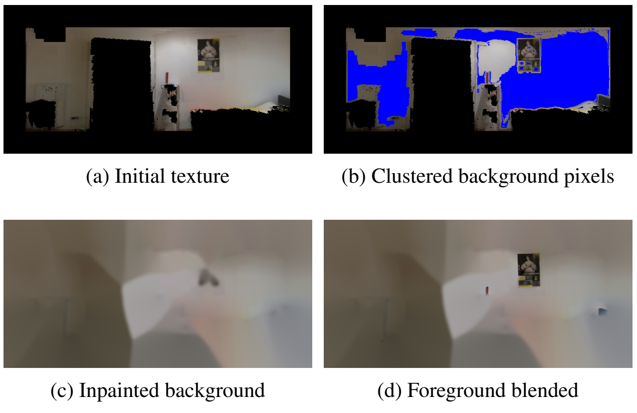
(a) Initial texture (b) Clustered background pixels (c) Inpainted background (d) Foreground blended

Figure 2: Per-plane texture generation step in Section 3.2.2.

$$E(\mathbf{T}) = E_g(\mathbf{T}) + \lambda_s E_s(\mathbf{T}) + \lambda_d E_d(\mathbf{T}) \tag{2}$$

The weights λ_s and λ_d are set 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:

$$E_g(\mathbf{T}) = \sum_i \sum_k \|\mathbf{p}_{i,k}^{plane} \cdot \mathbf{e}_3\|^2, \tag{3}$$

where $\mathbf{e}_3$ represents the unit-vector in the 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 $(\mathbf{p}_{i,k}^{plane}, \mathbf{p}_{j,k}^{plane})$ for every pair of (i, j) frames:

$$E_s(\mathbf{T}) = \sum_{i,j} \sum_k \|(\mathbf{p}_{i,k}^{plane} - \mathbf{p}_{j,k}^{plane}) \cdot (\mathbf{e}_1 + \mathbf{e}_2)\|^2. \tag{4}$$

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

$$E_d(\mathbf{T}) = \sum_i \sum_k \|C(\mathbf{p}_{i,k}^{plane}) - I_i(\mathbf{p}_{i,k}^{plane})\|^2. \tag{5}$$

We use the Ceres Solver (http://ceres-solver.org/) to optimize the registration. For the dense term and geometric term, we use only one point for every five 5 pixels in both x and y directions to reduce the problem size.

3.2.2. Foreground-background optimization

Even though we resort to simple planar geometry, we can still create the illusion of a realistic environment by rendering the model with high-resolution texture. For each pixel on the texture plane, multiple measurements are combined using a weighted median considering both geometry and color. However, combining measurements is not enough to create high-resolution texture. There are still possible misalignments due to geometric errors or motion blur, and, more importantly, unknown texture of the generated geometry extrapolated using room-structure priors.

We take a two-step approach for background and foreground. The underlying assumption is that each individual plane has a dominant color (base texture or background color) that can be smoothly interpolated and filled. On the other hand, there are high-frequency details on top of the base texture, which are assigned as foreground. The texture refinement steps for the foreground and background regions are described in Figure 2.

An example of the combined initial texture is shown in Figure 2a. We first cluster the RGB values of the pixels in the generated texture avoiding the edge region as shown in Figure 2b. We use the selected pixels to create the inpainted background (Figure 2c). The inpainted background region is merged with the initial texture to create a full texture without missing values (Figure 2d). Our approach with background assumption works with most flat walls in the real world. More importantly, the background regions are represented by a simple reflectance model and can be used to estimate the necessary light parameters.

4. Light parameter estimation

We solve for the light parameters using the homogeneous background region under the assumption of planar geometry. We are able to create full 3D content that contains not only geometry but also reflectance and light parameters representing both direct and indirect lighting. 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 the light parameter estimation of previous methods which either focused on small regions to place virtual objects, or considered only directions to represent an environment map. With fuller components that represent the environment, we can seamlessly augment virtual objects, create realistic shadows, and change texture or lights.

Let us first begin with the famous radiance equation [CWH93]:

$$R(V \to x') = \int_{x \in S} f(x \to V \to x') G(V,x) R(x \to V) dA. \quad (6)$$

The radiosity of a point V to another point x' is the sum of all radiance received from other points x multiplied by the BRDF f and the visibility G. We further assume that the 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 above equation can be re-written for a discretized mesh as

$$\begin{aligned} C(\mathbf{p}_k) &= \rho \sum_j G(\mathbf{p}_k, L_j) R(L_j \to \mathbf{p}_k) \\ &= \rho \{x(\mathbf{p}_k) + \sum_j D(L_j \to \mathbf{p}_k)\}. \end{aligned} \quad (7)$$

In other words, the pixel intensity $C(\mathbf{p}_k)$ is a combination of the direct lighting $\sum_j D(L_j \to \mathbf{p}_k)$ and the indirect lighting $x(\mathbf{p}_k)$.

The locations of the direct light are detected as the bright regions when the vertex colors are equalized as described in Section 3.1. For each detected location, we placed point-light sources with axially symmetric distribution with the vertical axis of symmetry represented by a 32-bin discretization of the angle θ_{L_j}. The occlusions are represented by a binary visibility function $G(\mathbf{p}_k, L_j)$.

To summarize, the direct lighting can be written as:

$$D(L_j \to \mathbf{p}_k) = G(\mathbf{p}_k, L_j) \cdot I_{L_j} \cdot A_{L_j}(\theta_{L_j}) \cdot \cos\theta_{\mathbf{p}_k} \cdot \frac{1}{r^2}. \quad (8)$$

We only need to solve for the light intensity, which is formulated as the combination of RGB factor I_{L_j} and angular 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 with the known information.

The physically correct way to simulate such lighting is to run ray tracing multiple times with the correct geometry and material, until reaching convergence. When converged, the field of indirect light can be represented by the position and the direction at any point within the volume. This involves a prohibitive amount of calculation and memory. Instead, we assign an unknown indirect lighting at each vertex $x(\mathbf{p}_k)$, and add smoothness criteria for neighboring vertices. In other words, we minimize the following optimization problem:

$$\sum_k \|C(\mathbf{p}_k) - \rho \cdot \{\sum_j D(L_j \to \mathbf{p}_k) + x(\mathbf{p}_k)\}\|^2 + \lambda_c \sum_{(k_k,k_2) \in N} \|x(\mathbf{p}_{k_1}) - x(\mathbf{p}_{k_2})\|^2 \quad (9)$$

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 use sample vertices on the plane region clustered as background.

5. Results

The pipeline has been implemented in a desktop machine with Intel Core i7 3.6GHz CPU with 16 GB memory. The rendering of virtual scenes is implemented using the Unreal engine (www.unrealengine.com) with point-light sources. We tested the pipeline with sequences available from [HDGN17], the BundleFusion [DNZ*17], and the ScanNet [DCS*17] data set. The intial mesh is built with the VoxelHashing approach [NZIS13]. The details of the dataset used is available in Table 1. After about 2-3 hours of processing, the complete and light weight representation is acquired with only 1-6% of face elements.

Figure 3 shows the qualitative comparison between the original volumetric reconstruction and the light-weight reconstruction. With a fraction of elements, we can still convey the overall shape of the environment. Samples of reduced triangle faces are highlighted in the bottom rows of Figure 3. More importantly, our pipeline greatly reduces the ghosting artifacts near the depth boundaries (shown in yellow) and fill unnecessary holes (shown in green). Most of details on texture is crisp, and the color stays equalized regardless of per-frame white-balancing. Our representation erases non-planar objects from the original reconstruction and fill them with the background color.

scenes	3dlite apt	BundleFusion office0	BundleFusion office3	ScanNet 0567_01
# of frames	2865	6159	3820	2066
faces (before)	3,291,072	926,414	3,045,096	435,126
# of planes	30	23	29	15
faces (after)	21,092	9,490	33,370	11,054
reduction	6.4%	1.0%	1.1%	2.5%

Table 1: The characteristics of indoor data sets and the representation processed by our pipeline.

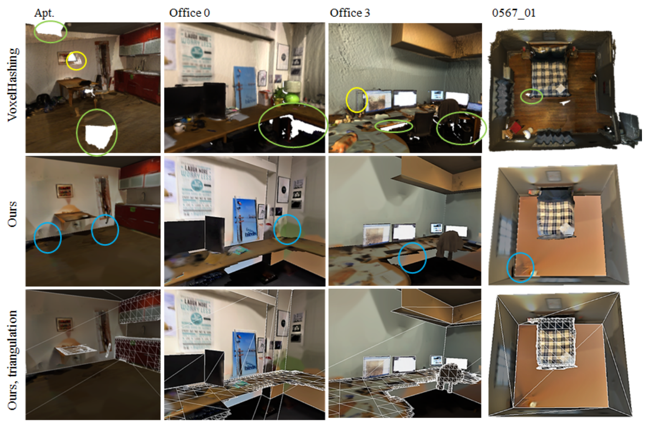

Figure 3: Comparison of visualization using VoxelHashing [NZIS13] (top rows) with our representation (middle and bottom rows). The volumetric reconstruction suffers from ghosting (yellow) or holes (green), and our approach alleviates the artifacts. In the meanwhile, our approach erases non-planar objects (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.

With the help of planar proxy, the visualization of different viewpoints stays convincing. We compare the original input frames (first column) with renderings of textured mesh in Figure 4. The simple rendering of transformed initial mesh reveals the limitation of imprecise texture (second column). Even with re-texturing of equalized frames with weighted median can alleviate such artifacts (third column). We then find the background colors of planes and use the light location to solve simplified inverse rendering problem. From the inferred values and using the simplified planar geometry, we can create a virtual scene of an empty room (forth column). The lighting and texture components might not be exact, but we can still render similar colors of planes. The virtual scene can be freely altered for VR applications (fifth column).

6. Conclusion

We presented a holistic pipeline to represent captured indoor environment into a 3D content with full geometry, texture, and lighting information. We first focus on completing the room structure based on plane detection. Individual planes are further refined detecting dominant colors for background. The detected backgrounds are used to fill unobserved region and extract components for inverse rendering, while the remaining foreground is used to refine the texture. The generated representation can be used to visualize and navigate the captured environment.

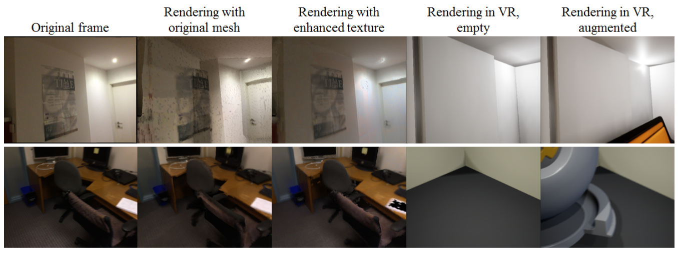
Original frame
Rendering with original mesh
Rendering with enhanced texture
Rendering in VR, empty
Rendering in VR, augmented

Figure 4: Samples of input frames and the rendering at the same view for apt and office0 data set.

Acknowledgements

This work was partly supported by the ICT R&D program of MSIP/IITP. [2017-0-00162, Development of Human-care Robot Technology for Aging Society] and KIST institutional program [Project No. 2E29450]

References

[CWH93] Cohen M. F., Wallace J., Hanrahan P.: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Inc., San Diego, CA, USA, 1993. 3

[DCS*17] Dai A., Chang A. X., Savva M., Halber M., Funkhouser T., Niessner M.: Scannet: Richly-annotated 3D reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017). 1, 3

[DNZ*17] Dai A., Niessner M., Zollöfer M., Izadi S., Theobalt C.: Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)* (2017). 1, 3

[FTK14] Feng C., Taguchi Y., Kamat V.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *Proceedings of IEEE International Conference on Robotics and Automation* (2014). 1

[HDGN17] Huang J., Dai A., Guibas L., Niessner M.: 3DLite: Towards commodity 3D scanning for content creation. *ACM Transactions on Graphics 2017 (TOG)* (2017). 1, 3

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), SIGGRAPH '87, pp. 163–169. 2

[MMBM15] Monszpart A., Mellado N., Brostow G. J., Mitra N. J.: Rapter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph. 34*, 4 (July 2015), 103:1–103:12. 1

[NZIS13] Niessner M., Zollhöfer M., Izadi S., Stamminger M.: Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)* (2013). 3, 4

[ZCC16] Zhang E., Cohen M. F., Curless B.: Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016) 35*, 6 (2016). 2