

# Structured Indoor Modeling

Satoshi Ikehata

Hang Yan

Yasutaka Furukawa

Washington University in St. Louis

## Abstract

*This paper presents a novel 3D modeling framework that reconstructs an indoor scene as a structured model from panorama RGBD images. A scene geometry is represented as a graph, where nodes correspond to structural elements such as rooms, walls, and objects. The approach devises a structure grammar that defines how a scene graph can be manipulated. The grammar then drives a principled new reconstruction algorithm, where the grammar rules are sequentially applied to recover a structured model. The paper also proposes a new room segmentation algorithm and an offset-map reconstruction algorithm that are used in the framework and can enforce architectural shape priors far beyond existing state-of-the-art. The structured scene representation enables a variety of novel applications, ranging from indoor scene visualization, automated floorplan generation, Inverse-CAD, and more. We have tested our framework and algorithms on six synthetic and five real datasets with qualitative and quantitative evaluations. The source code and the data are available at the project website [15].*

## 1. Introduction

Indoor scenes exhibit rich geometric and functional structures, which are carefully designed to optimize the quality of our private, social and economic activities. A fundamental task in indoor modeling is to discover structural elements constituting an indoor scene, such as rooms, doors, and walls, then reconstructs a structured 3D model consisting of such elements. The potential applications of such 3D models range from architecture, civil engineering, digital mapping, urban geography, real estate, and more.

Indoor scene understanding and reconstruction has been an active research topic in computer vision. However, existing work has focused on small-scale indoor scenes such as a single room or a corner of a single room [11, 24]. Semantic reconstruction (SR) has been active in Computer Vision and Robotics [14], but their focus is again small-scale and/or on clutter-analysis. Furthermore, they assign semantics to an existing geometry, as opposed to leverage the semantics for reconstruction. Building-scale indoor model-

ing approaches exist. However, their output is either a pure polygon soup [30] or a set of planar patches [31].

We establish a computational framework and algorithms for reconstructing *structured indoor model* from panorama RGBD images. We introduce a novel 3D model representation “structure graph”, whose nodes represent structural elements such as rooms, doors, and objects, and the edges represent their geometric relationships. “Structure grammar” then defines a list of possible graph transformations. This grammar drives a principled new reconstruction algorithm, where the rules are sequentially applied to naturally segment, annotate, and reconstruct architectural scenes.

The framework allows one to design a different geometric representation (*e.g.*, mesh, depthmap, or point-cloud) and a different reconstruction algorithm to enforce the most effective architectural prior for each structural element. An effective choice is possible, since our approach is top-down and the full context is given in reconstructing each element.

The paper also proposes new room segmentation and offset-map reconstruction algorithms that produce high quality yet extremely compact 3D models. Despite the flexible representation, our framework guarantees to produce a manifold mesh, which enables a wide range of new applications. Our framework is evaluated qualitatively and quantitatively with both the synthetic and real data.

In summary, the contributions of the paper are three-fold:

**Framework contribution:** A principled new structured model representation and a reconstruction framework. This framework is general and applicable to other domains, such as outdoor architectural scenes or man-made objects, where the subjects consist of structural elements with regularities.

**Technical contribution:** A room segmentation and reconstruction algorithm that classifies room connection types, and an offset-map reconstruction algorithm that produces extremely compact model by enforcing both the low-entropy and the low-rank structure in the output label space.

**System contribution:** This paper is the first to 1) produce segmented and annotated models from the level of rooms to objects, 2) generate floorplans automatically, and 3) enable a compelling Inverse-CAD application for indoor scenes.

## 2. Related Work

This paper touches many topics in diverse fields from Computer Vision, Computer Graphics and Robotics. We here limit the discussion to closely related topics.

**Small-scale indoor modeling:** Indoor scene modeling has been active in Computer Vision, where a room layout and/or object placements are inferred from an image [13, 5], an RGBD image [11, 16]. Data driven approaches with 3D model databases yield CAD-quality reconstructions [19, 22]. However, the focus of these methods is so far on a clutter analysis in a small scale such as a single room. Semantic reconstruction (SR) from an RGBD stream has been active in Robotics and Computer Vision [17, 14]. These methods handle slightly larger scenes, but are still not at the scale of an entire scene with multiple rooms. Also, SR merely assigns semantics to the pre-reconstructed 3D models. Segmentation, recognition, and reconstruction are unified into a single formulation in [12], but their operating range is also small, as the entire scene is modeled as a voxel grid.

**Large-scale indoor modeling:** Recent developments in the sensor technology enable us to capture dense 3D points in a large scale. However, when it comes to the construction of a real 3D model (e.g., a mesh), existing methods only produce a set of planar patches at a room scale [31], simple primitives for a part of a scene [10], a dense mesh from a voxel grid [29], an extruded model from a floorplan [29], or a “polygon soup” without any structure or semantics [30]. For large scenes, room segmentation is also an important problem. The current state-of-the-art methods solve in a top-down 2D domain [29, 21], although room boundaries are often ambiguous in a top-down view. We explicitly classify room connection types with full 3D information.

**Architectural shape priors:** Primitive detection is combined with Markov Random Field (MRF) to reconstruct piecewise planar depthmaps [6]. Detected 3D primitives can be assembled directly in the 3D space [30]. However, excessive number of primitives need to be extracted in general not to miss a single important one, and the primitive detection often becomes the most problematic step. Instead, we combine MRF with Robust Principal Component Analysis (RPCA) [3] to enforce both the low-entropy and the low-rank structure in the depthmap to obtain much more compact 3D models without extracting any primitive.

## 3. Structured Modeling

The key innovation lies in the structured representation of scene geometries, and its reconstruction framework. We represent a scene as a “structure graph” and conduct reconstruction as a sequence of “structure grammar” applications. This section presents these key concepts (See Fig. 1).

**Structure graph:** A scene geometry is represented as a

graph, where nodes correspond to structural elements such as a room, a wall, and an object. Each node is associated with geometry (e.g., a mesh, a depthmap, or voxels) with the exception of “scene” and “room”, which are abstract elements. Our model works with any geometric representation, and even allows different representations for different elements (e.g., a solid for a room and a mesh for an object).

Nodes are connected by three types of edges. First, when structural elements have a common surface boundary (e.g., adjacent wall nodes), the nodes are connected by a solid undirected edge (purple in Fig. 1), enforcing that the geometries match exactly along the boundary. This geometric consistency, dubbed “boundary condition”, is the key to guarantee manifold-ness later. A dashed undirected edge, on the other hand, represents an attachment relationship without the boundary constraint (e.g., an object in a room).<sup>1</sup> Lastly, a directed edge encodes the level-of-detail relationship: Structural elements at the children nodes constitute a detailed version of the element at their parent. For example in Fig. 1, a “Wall” node contains a simple quad surface. The child node “Wall w/ hole” contains a hole in the middle of the quad to represent the doorway. Note that the boundary condition of a parent must be also satisfied by the children nodes to guarantee the manifold-ness.

**Structure grammar:** Structure grammar defines a set of transformation rules that are allowed on a structure graph, where each rule application triggers a reconstruction algorithm. A rule has two components. 1) A “pre-condition” must be satisfied by the current graph to be applicable. 2) The “transformation” describes how the structure graph should change. Take a room reconstruction for example. The rule takes a room node (pre-condition). The rule produces a floor node, a ceiling node (not shown to avoid clutter), and a chain of wall nodes (transformation). The details of the rule set are given in Sect. 4.

**Structured reconstruction:** The grammar drives our structured reconstruction process, where the grammar rules are sequentially applied to recover a structure graph together with the geometries. For each rule application, we choose a geometric representation and a reconstruction algorithm that are suitable for the task. For example, the room reconstruction rule recovers a 1D polygon shape and extrudes it to the floor and the ceiling heights to reconstruct a room model. The 1D polygon is obtained by a special algorithm that is designed to produce a piecewise planar compact polygon [2]. Two types of geometric constraints need to be enforced in the reconstruction process. First, boundary conditions must be satisfied: 1) Existing boundary conditions (i.e., undirected solid edges) before the rule application must be preserved; 2) New boundary conditions must

<sup>1</sup>We attach objects to room nodes. Give more precise contact relationships, the framework allows one to attach to other nodes easily (e.g., floor).

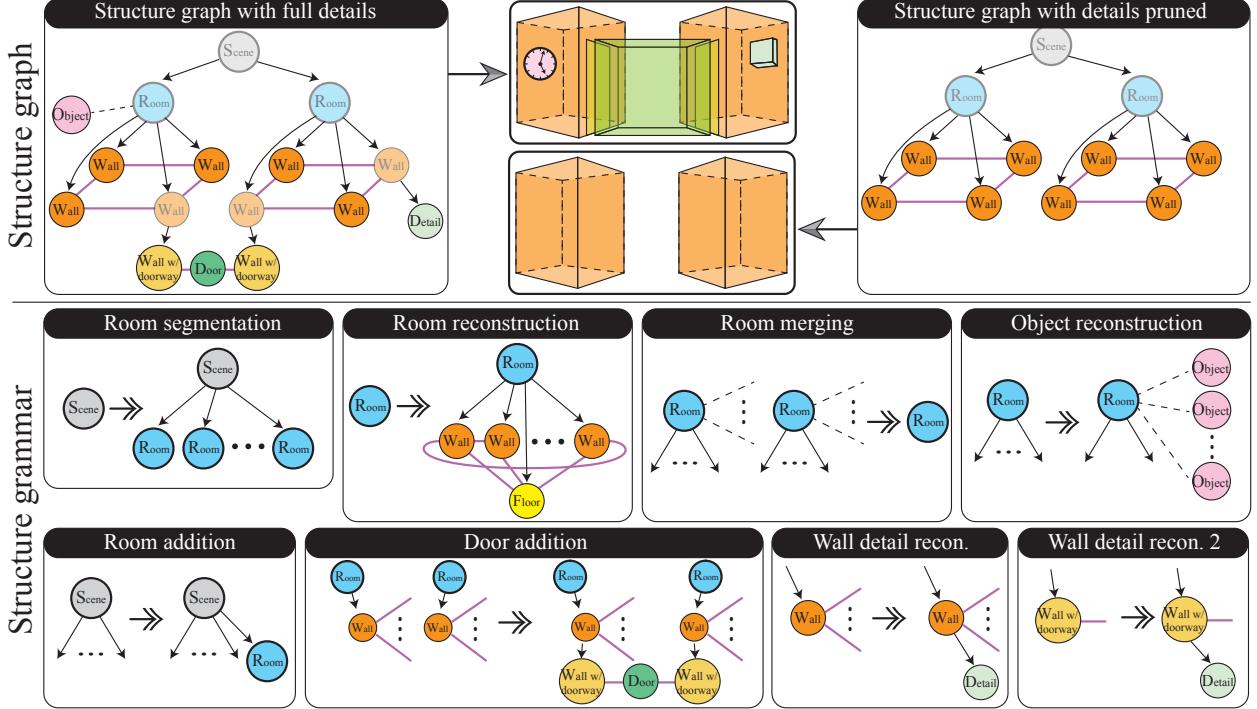


Figure 1. Top left: An indoor scene is modeled as a *structure graph*, where nodes correspond to structural elements such as a room, a door, or an object. Each node is associated with a geometry such as a mesh or a depthmap, except for the scene and the room nodes. Edges contain their geometric relationships. A mesh model can be generated from the graph by outputting a mesh from every leaf node (a node without an out-going directed edge). Top right: One can drop an arbitrary set of nodes from the leaves for the mesh generation. As long as no dangling solid undirected edges remain, the mesh is guaranteed to be a manifold. Bottom: *Structure grammar* defines a set of possible transformations of this graph. Our reconstruction process is to sequentially apply these rules to recover the structure graph.

be satisfied. Second, reconstructed geometries must not intersect with the existing geometries in the other nodes. Although these constraints may appear very complex, it is fairly straightforward to enforce them, as shown in Sect. 4.

**Mesh compilation:** While the structure graph is flexible and allows different geometric representations at different nodes, it is often desirable to produce a mesh model for applications. The graph allows one to compile a manifold mesh by simply producing the mesh geometry from each leaf node (i.e., a node without an out-going directed edge). In the top left of Fig. 1, non-leaf nodes are grayed-out. Here, the geometry at each node needs to be converted to a mesh if necessary, which is easy for standard geometric representations (e.g., a depthmap by simple meshing, a volumetric scalar field by the Marching-Cube method [20], or a point cloud by Poisson Surface Reconstruction [23]). Furthermore, complexity and details of the mesh model can be easily controlled by dropping an arbitrary set of nodes from the leaves. Since the boundary constraint at any node is satisfied by the children, we can inductively prove that the compiled mesh model would also be a manifold, as long as no dangling solid undirected edges remain. Please see the supplementary material for the full proof and theory.

**Assumptions:** We assume that a scene has a single story and that the room structure (i.e., floor, ceiling, and walls) is aligned with the Manhattan directions. However, this restriction is due to our particular choice of a structure grammar and reconstruction algorithms. One can certainly change the grammar and algorithms to extend the capabilities (e.g., multi-story and non-Manhattan buildings).

## 4. Structured Indoor Modeling

This section presents our implementation of the structured modeling framework for the indoor scenes. After explaining our input data, we will show the overall modeling framework and individual structure grammar rules with the corresponding reconstruction algorithms.

### 4.1. Input data

Panorama RGBD images are acquired by a camera and a depth sensor mounted on a motorized tripod. It takes a couple of minutes to acquire a single panorama RGBD image, and they are aligned by the ICP algorithm after rough manual initialization. Several filtering techniques are used to pre-process data, whose details are referred to the supplementary material. There are a few things worth noting

here. First, we extract the Manhattan directions from the input point clouds [6], and transform the data into the Manhattan coordinate frame with the Z-axis pointing to the “up direction”. Second, the point  $P(v)$  and the free-space  $F(v)$  evidences are calculated for each voxel  $v$ . We discretize the bounding box of the 3D points, where the voxel size is set to 0.012m.  $P(v)$  counts the number of 3D points inside.  $F(v)$  counts how many times the voxel is intersected with visible rays.  $P(v)$  and  $F(v)$  are normalized so that their greatest value becomes 1, respectively.

## 4.2. Modeling pipeline

The structured modeling repeats applying grammar rules whose pre-conditions are satisfied, until no rules can be applied. The structure graph is initialized with a scene node, and the room segmentation is the only applicable rule initially. In practice, the following three guidances are also used to control the rule applications: First, the wall detail reconstruction is applicable only after the system terminates with the remaining rules. Without the restriction, the wall details might be reconstructed for a room, but the room could be later merged and removed, yielding wasted computations. Second, whenever a new room node is created, the room reconstruction rule is triggered, as this is the only applicable rule for a newly generated room node. Lastly, the process converges in practice (See Fig. 2) but is not guaranteed to terminate as it may iterate creating and removing rooms. The room merging rule is applied at most once.

## 4.3. Indoor structure grammar

Our indoor structure grammar consists of the eight rules in Fig. 1. We restrict our description to the major rules here. The remaining three rules together with minor algorithmic details are given in the supplementary material. The example progress of the graph construction is shown in Fig. 2.

**Room segmentation:** The rule obtains the initial room segmentation on the XY-plane. As room boundaries are often ambiguous in 2D, 3D analysis will be conducted in future rules to refine the segmentation. The rule takes a scene (pre-condition) and creates multiple rooms (transformation).

The room segmentation is formulated as a clustering problem. First, the domain  $\Psi$  is initialized by the pixels, where the sum of the free-space evidences along the Z-axis is non-zero.  $\Psi$  is then refined by removing pixels, whose distances to the closest boundary of  $\Psi$  along the X or Y axes are less than 0.8m. This heuristic is effective in removing thin erroneous regions near windows. Second, the k-medoids algorithm is used to cluster subsampled pixels (at every 100mm along X and Y) in  $\Psi$ . The distance metric for clustering is based on a “binary visibility vector”  $V(p)$  for each pixel  $p$ . The  $i_{th}$  element  $V_i(p)$  of the vector is 1, if  $p$  is visible from the  $i_{th}$  boundary pixel through  $\Psi$ . Intuitively, the vector  $V(p)$  stores which of the scene boundary

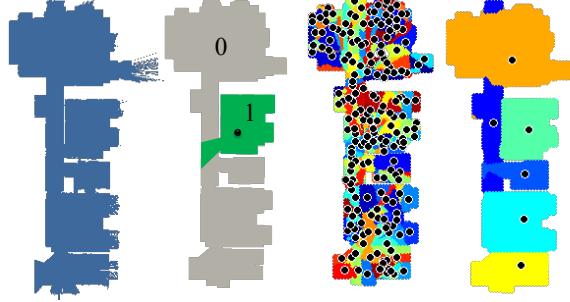


Figure 3. Room segmentation. From left to right: 1) The domain is obtained by thresholding the free-space evidence. 2) The refined domain, and the visibility information at a pixel. 3) Initial 200 clusters of the k-medoids algorithm. 4) The final clusters. A pixel in the domain is colored based on the nearest cluster center.

is visible from  $p$ . The distance of the two pixels  $p$  and  $q$  is given by the hamming distance of  $V(p)$  and  $V(q)$  divided by  $\sum_i V_i(p) \sum_i V_i(q)$ . The division converts the distance range to  $[0, 1]$ . Starting from 200 clusters, we repeat the k-medoids algorithm and the cluster merging, where two adjacent clusters are merged if the distance between their centers is less than 0.4. Lastly, the segmentation results are propagated to all the pixels in  $\Psi$  by the nearest neighbor.

**Room reconstruction:** This rule takes a room that has not been reconstructed yet, that is, a room node with only incoming directed edges (pre-condition). A room outline in a top-down view is reconstructed as a 1D polygon, which is extruded to the estimated floor and ceiling heights to generate a 3D model. This rule generates a floor, a ceiling and a chain of wall nodes (transformation).

A room outline is generally piecewise planar and consists of a very few number of vertices. We employ the shortest-path based algorithm that is optimized to produce such shapes [2]. There is one modification worth noting. The original algorithm requires the addition of “anchor” points to overcome the shrinkage bias, because they reconstruct an entire scene (i.e., multiple rooms) with a single shortest-path algorithm. In this work, the algorithm is applied to each segmented room individually, and does not require anchor points. The floor and ceiling heights are estimated by horizontal plane fitting via RANSAC, below and above the average camera height, respectively.

**Wall/ceiling detail reconstruction:** This rule recovers wall details such as windows or counters as an “offset-map”. The rule takes a wall or a wall with a doorway (pre-condition), and generates a new node with more detailed geometry under the same boundary conditions (transformation). The same goes for the ceiling. The wall details are represented as an axis-aligned 2D array of offset values along the wall normal. Where the size of each pixel is 0.012 m. The offset value is zero, positive, and negative if the structure is on, in front of, or behind the wall, respectively.

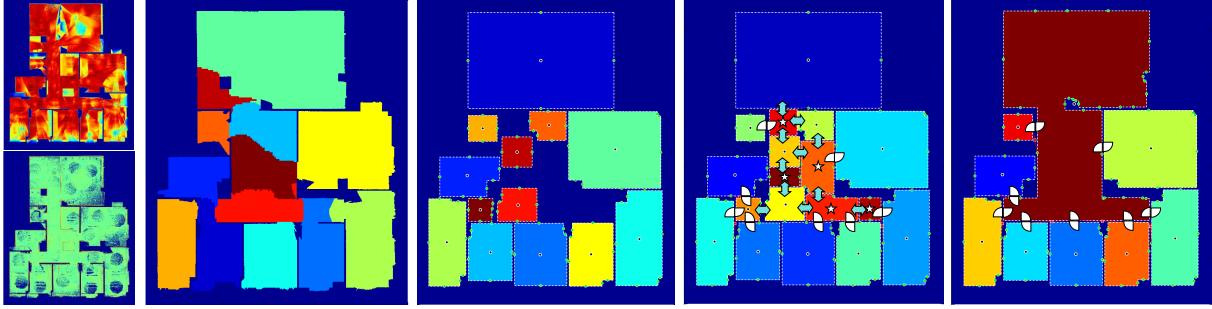


Figure 2. Structured graph representation and its construction. Starting from the generation of the free-space end point evidence (left), we perform the room-segmentation and reconstruction (2-nd and 3-rd column). More rooms are added and the room connection types are classified (4-th column). The white icon represents the “door”, while the blue icon represents the “merge” classification. We finally get the structured graph representation as shown in the last column. Here we only show the spatial relationships for simplicity.

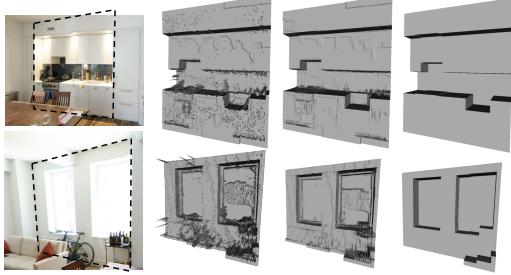


Figure 4. Wall detail reconstruction. From left to right: 1) an image; 2)  $D_m$  at the initialization; 3)  $D_r$  after the RPCA optimization; and 4)  $D_m$  after the MRF (with the label cost) optimization.

Piecewise planar depthmap algorithms set up a problem so that the low-entropy (a few labels) solutions correspond to piecewise planar depthmaps [6, 26, 8]. This prior is also used in our formulation. A key difference is that our offset-map is defined on an axis-aligned wall, and the label boundary (*i.e.*, depth discontinuity) is also axis-aligned. The enforcement of piecewise planar “label boundary” is a more challenging problem. We are aware of only one work [25], which effectively enforces this type of constraint for binary images. However, line segments must be extracted as the label boundary candidates, and the running time is exponential in its number. Our key observation is that an axis-aligned “label boundary” means low-rank, where we have an effective optimization tool such as Robust Principal Component Analysis (RPCA) [3] to enforce this constraint.

More specifically, we first initialize the offset-map by a standard MRF with unary and pairwise terms (See the supplementary material for the energy definitions). We then repeat solving the following two problems:

$$\begin{aligned} \min_{D_r} \|E\|_1 + \mu_1 \|D_r\|_* &\quad s.t. \quad D_m = D_r + E, \\ \min_{D_m} \|D_m - D_r\|_2^2 + \mu_2 \|\nabla D_m\|_0 + \mu_3 \text{Label}(D_m). & \end{aligned} \quad (1)$$

The first problem decomposes the input offset-map  $D_m$

into a low-rank matrix  $D_r$  and a sparse error matrix  $E$  via RPCA. The second problem takes the low-rank matrix  $D_r$ , and enforces low-entropy via MRF. The first term in the second problem is a unary term ensuring the consistency between  $D_r$  and  $D_m$ , the second term is a simple Potts model penalizing the label changes, and the third term is the label cost [4] further enforcing the low-entropy structure. The iteration converges very quickly in general (2 to 3 iterations). The final offset-map is set to  $D_m$ , which is compact and requires few polygons for meshing (See Fig. 4).  $\mu_1$  is set to  $\sqrt{\max(W, H)}$  as in [3], where  $W$  and  $H$  are the dimension of the offset-map in pixels.  $\mu_2$  and  $\mu_3$  are set to 500 and  $10^5$ , respectively.

**Object reconstruction:** The rule takes a room, whose shape has been reconstructed, that is, a room node with outgoing directed edges (pre-condition), then segments the remaining unexplained 3D points, and generates object nodes (transformation). Note that a point-cloud is used as the geometric representation, as the scene visualization is the key application and a point-based rendering is suitable for incomplete objects rather than a mesh.

Given a room node and its extruded 3D model, we first collect 3D points that are inside the room but not too close to the existing geometry with a margin of 0.1 times the room height. A variant of region growing segmentation algorithm in PCL [1] is used to segment the point-clouds<sup>2</sup>. For better rendering effects, we remove small objects that contain less than 1000 points, as well as floating objects whose minimum Z-coordinate is above the room center.

**Room connection classification:** Pre-conditions of “door-addition” and “room-merging” depend on the classification of the room connection types, which are either “door”, “merge”, or “no-connection”. The first and the second classifications are the pre-conditions of the door-addition and room-merging rules, respectively. This classifier is the key

<sup>2</sup>Note that the indoor object reconstruction is an active research topic and our framework allows one to easily plug-in new algorithms.

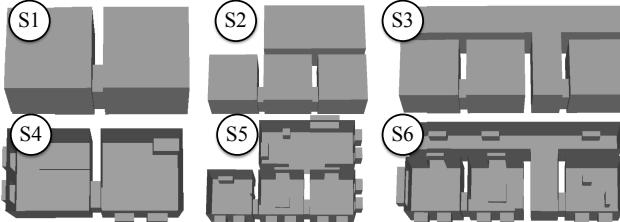


Figure 5. Synthetic datasets. RGB-D panoramas are generated from sparsely distributed viewpoints from these meshes.

to improve the final room segmentation quality.

Given a pair of rooms, we identify a pair of walls that are parallel, are within a distance of 1.0 m along the normal, and have non-zero lateral overlap. For such pair of walls, we compute the most-likely shape of the door on the walls based on the nearby point and free-space evidences. More concretely, each wall is discretized into an array of pixels, whose size is the same as the voxel (*i.e.*, 0.012m). The point  $P_w(p)$  and free-space  $F_w(p)$  evidences on the wall are obtained by summing  $P(v)$  and  $F(v)$  along the normal direction with a margin of 5 voxels, respectively.  $P_w$  and  $F_w$  are calculated for each of the two walls. With abuse of notation,  $P_w(p)$  and  $F_w(p)$  denote the sum over the two walls for each pixel  $p$ . We want  $P_w$  and  $F_w$  to be small and large inside the doorway, respectively. Therefore, the door shape is obtained by finding the rectangle  $\Omega$  that minimizes

$$\sum_{p \in \Omega} P_w(p) - \lambda_1 \sum_{p \in \Omega} F_w(p) + \lambda_2 \{\text{area of } \Omega\}, \quad (2)$$

where  $\lambda_1$  and  $\lambda_2$  are set to 0.01 and 0.001. The last term is to prefer a smaller rectangle when the solution is ambiguous due to the lack of evidence. This can be efficiently computed by using the integral images. Given the detected rectangle shape and the pair of walls, the connection type is classified. If the rectangle is too small (the longest dimension is less than 0.6 m), it is classified as “no-connection”. The classification between “door” and “merge” are based on pre-determined thresholds. For example, if the width and height of the walls are different from those of the door with a margin, the connection becomes “door”. Possible configurations are numerous, and the full classification table is given in the supplementary material.

## 5. Experimental Results

The proposed system has been evaluated on five synthetic and six real datasets. All the experiments have been performed on an Intel Core i7-4770 (3.40GHz, single thread) machine with 32.0GB RAM. MATLAB with C++ mex functions are used for the implementation.

**Synthetic data:** We generate synthetic indoor 3D models as shown in Fig. 5. The first three datasets (S1)-(S3) contain multiple rooms (2 or 4) but without any details on the walls,

Table 1. Positional [mm] and normal errors [degrees] (inside parentheses) on synthetic data.

	Ours w/o Obj.	Ours w/ Obj.	Poisson (high-res)	Poisson (low-res)	Vgcuts (high-res)	Vgcuts (low-res)
S1	12.5 (1.3)	<b>5.8 (1.3)</b>	15.0 (1.8)	15.0 (1.6)	32.0 (3.5)	35.0 (3.2)
S2	20.4 (1.7)	<b>8.6 (1.7)</b>	23.0 (2.3)	26.0 (2.5)	16.0 (2.1)	22.0 (2.2)
S3	43.1 (2.4)	<b>42.0 (2.4)</b>	43.0 (3.1)	43.0 (2.8)	67.0 (6.7)	64.0 (6.0)
S4	23.5 (3.2)	<b>8.5 (1.9)</b>	17.0 (2.6)	18.0 (2.5)	17.0 (3.0)	24.0 (3.3)
S5	31.2 (2.2)	<b>19.0 (2.2)</b>	24.0 (3.0)	25.0 (2.7)	32.0 (4.3)	34.0 (4.2)
S6	52.0 (2.9)	<b>32.0 (2.6)</b>	33.0 (2.6)	39.0 (4.2)	37.0 (3.3)	

floors or ceilings. The latter three datasets (S4)-(S5) have the same rooms but contain objects and wall details.

We compare our algorithm against the baseline methods in Computer Graphics and Computer Vision, namely, the screened poisson reconstruction (Poisson) [18] and the Manhattan volumetric graphcuts (Vgcuts) [7]. We are interested in the trade-off between the geometric accuracy and the model complexity (*e.g.*, the number of polygons in the model). Therefore, a standard mesh simplification algorithm [27] is used to control the numbers of polygons for Poisson. For Vgcuts, we tune the weight on the smoothness penalty to control the polygon counts. For both methods, we generated a low-res and high-res mesh models. The polygon count of the low-res model is set to be very close to ours. High-res models by Poisson and Vgcuts usually contain  $10^5$  and  $10^2$  times more polygons than ours. Each mesh model is rasterized into a panorama image to generate a depth image together with the surface normal information.

The positional and the normal accuracies are measured simply by the average error from the ground truth over the rasterized pixels (See Table 1). We observe that even without the object point clouds, our method achieves high precision despite the low-polygon counts. Errors are mainly due to the lack of objects, which can be verified by our results with object point clouds, where point clouds are rasterized with splatting. Poisson often overfits the surface on the dense point cloud and the normals on the surface are cluttered. Vgcuts has a strong axis-aligned regularization, and tends to lose details due to the shrinkage bias.

**Real data:** Figure 6 illustrates the five real datasets, where their statistics are summarized in Table 2. Unlike the synthetic datasets, 3D points in the real datasets are corrupted by mirrors, windows, reflective objects, and calibration errors. The second row in Fig. 6 shows the reconstructed meshes, compiled from the leaf nodes except for objects in the structure graph. The color of each geometry is determined by the parent room node. The third row shows the full renderings including the object point clouds, which are rendered by the point-based rendering technique. Our models are segmented and annotated with the structural element types (*e.g.*, floor and walls), and the ceilings are not rendered on purpose to visualize the interior, which is not easy for standard reconstruction methods such as Poisson

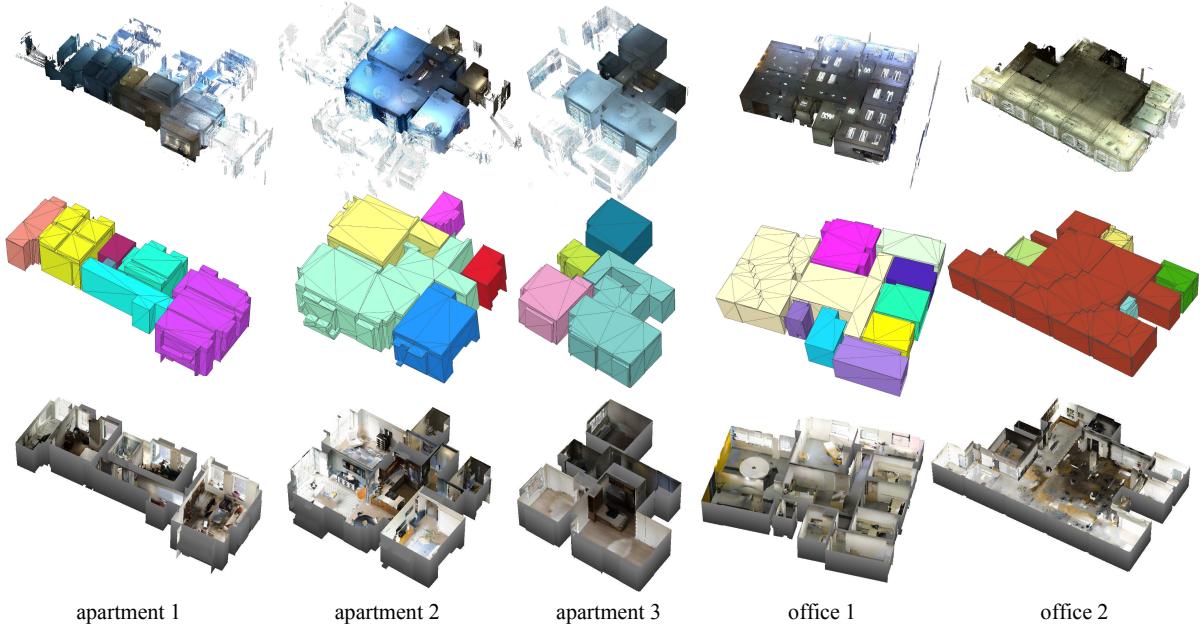


Figure 6. Results on the real datasets. From top to bottom: the input point clouds, the reconstructed mesh models, and sample renderings.

Table 2. Statistics of the real experiments. Here we show the number of elements reconstructed and the geometric distance from the input depth map and reconstructed surface geometry compiled with (a) walls, (b) walls+details, (c) walls+details+objects. We also show the computational time of the entire pipeline.

Dataset	# panos	# points	Num. elements				Total dist. Error (mm)		
			Room	Wall	Obj.	Door	Wall	+Detail	+Object
Apart. 1	16	4933172	6	62	15	5	139	103	81
Apart. 2	15	10065236	5	63	24	4	181	122	86
Apart. 3	14	4534136	4	58	1	3	109	96	65
Office 1	33	4227235	9	113	19	8	155	145	46
Office 2	35	10225793	6	88	33	7	266	253	125

Dataset	Computational time (sec)				
	Initialization	Room Segmentation	Room, Floor, Ceil Recon.	Detail Recon.	Model Compilation
Apart. 1	62	86	682	1034	0.5
Apart. 2	48	68	468	558	0.8
Apart. 3	45	55	629	442	1.0
Office 1	109	255	1430	1068	1.3
Office 2	53	253	1603	1289	2.0

and Vgcuts. Although it is difficult to conduct quantitative evaluations on real datasets, Table 2 provides positional errors (the same metric as in Table 1) of our model with and without the wall details and the object point clouds. The table shows that our models achieve 10 to 20 cm average accuracy even without the object point clouds. More evaluations including comparisons against Poisson and Vgcuts are provided in the supplementary.

Figure 7 qualitatively compares our reconstructions against Poisson and Vgcuts. Again, the mesh simplification algorithm [27] and the parameter tuning is used to make their polygon counts close to ours. The figure shows that both Poisson and Vgcuts models contain clutters. Instead, our approach successfully ignores clutters in modeling architectural components, yielding clean and compact build-

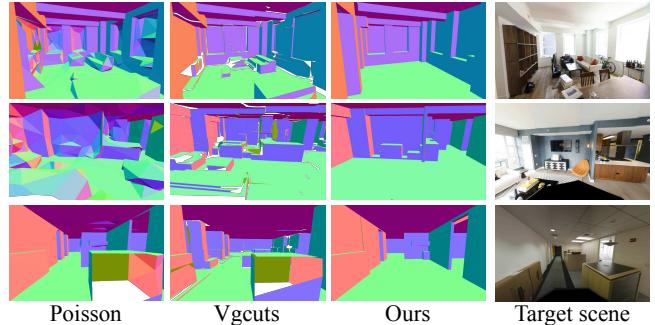


Figure 7. Comparison with the same number of triangular meshes. Here we show the result of (a)Poisson, (b) Vgcuts, and (c) ours. Note that we simplify the original meshes of Poisson and Vgcuts so that the models contain the same number of polygons with ours.

ing structure. It is worth mentioning that our wall detail reconstruction algorithm recovers interesting details on walls (*e.g.*, kitchen counter, window frame). These clean models enable high quality texture mapping and rendering experiences as well as interesting post geometry processing, as demonstrated in our supplementary video.

We compare our room segmentation process against two state-of-the-art methods [21, 29] in Fig. 8. We asked the authors to process our data with their native implementations. Our method provides reasonable segmentation that mostly preserves the number of structural cluster (*e.g.*, rooms and aisles) and important details. The over-segmentation is often observed in [29], and the room shape is too simplified and important details often disappear in [21]. Note that the results could not be obtained for three datasets by [21], mainly due to the failures in the primitive extraction step.

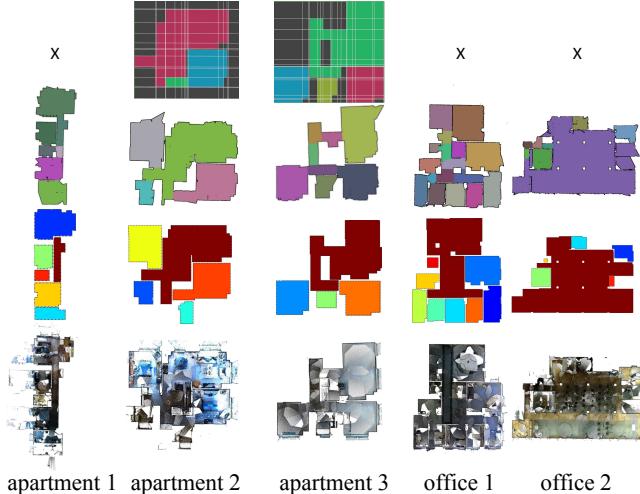


Figure 8. Comparison of the room segmentation algorithms. The first and second rows are the results by [21] and [29], respectively. The third rows contain ours. The bottom rows show the input point clouds for reference.

## 6. Applications

The structured scene representation facilitates novel applications. In this paper, we demonstrate the following four. Please also see our supplementary video.

**Floorplan generation:** The first application is the generation of floorplan images. One missing information is the types of objects and rooms, which are obtained by state-of-the-art object recognition software [32] and scene recognition software [9]. Experimental details and evaluations are in the supplementary material. In addition to placing annotation texts, we can automatically change the rendering style of each room and object depending on its type.

**Indoor scene viewer:** Our model enables a novel indoor scene viewer, which renders an indoor scene from ground to air, shows the floorplan image, displays room thumbnails, or highlights room structures. Room elements serve as interactive icons to enable a room-based navigation.

**Inverse-CAD:** The mesh model compiled from the structure graph is compact, yet regularized, and a manifold. We can export our model to a CAD system (*e.g.* Sketchup [28] as COLLADA format) for users to easily fix mistakes, add new structures, or clone new rooms within a minute or two.

**Tunable reconstruction:** As mobile applications dominate, it becomes crucial to intuitively and directly control the complexity of a model, because the computational powers of mobile devices vary but are limited. The structure graph allows us to, for example, specify the maximum number of polygons allowed for an entire scene (X), each room (Y), and each wall (Z). These constraints can be easily enforced by repeatedly dropping leaf nodes until the constraints are satisfied (See Fig. 9).

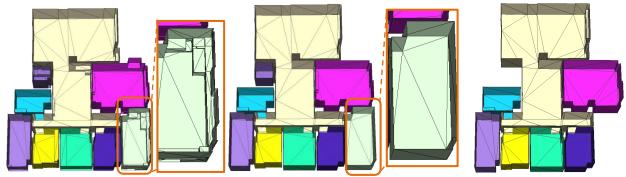


Figure 9. Structured model representation allows one to directly specify the maximum number of polygons allowed for an entire scene (X), each room (Y), and each wall (Z). The values of  $(X, Y, Z)$  are  $(2000, 500, 100)$ ,  $(1000, 500, 2)$ , and  $(500, 100, 10)$  from left to right in the figure. When the budget is really tight (right), the system might drop a few rooms.

## 7. Limitations and Future Directions

We admit that our current system has limitations. The most problematic step is the connection classification. Our current approach relies only on local geometric information, and tends to under-segment. A future work is to exploit images to robustify the classification process. Scene and object recognition accuracies are also poor at many places, where building-scale context from the structure graph may help. For example, a bedroom is unlikely to be in the middle of offices. Another future work is to extend current structure grammars to handle more sophisticated building structures *e.g.*, non-Manhattan or multi-story buildings. It could not be trivial but possible because of the high flexibility of our representation. We are also interested in testing our framework on RGBD stream datasets as opposed to panorama RGBD images. Although the camera pose quality tends to degrade for RGBD streams, our framework should be applicable.

We hope that this paper opens up new exciting research directions such as follows. Please visit our project website [15] for more information, source code, and datasets.

**Structured X modeling:** Structured modeling is a general framework, and is applicable to many other domains such as outdoor architectural scenes or man-made objects.

**Indoor Inverse Procedural Modeling:** Structure grammar is manually constructed in our experiments. An interesting alternative is to learn the grammar rules from examples.

**Inverse-CAD and Scan2BIM:** Inverse-CAD has been a long-standing research milestone in Computer Vision, Computer Graphics, and Civil Engineering. For major architectural components, our models already enable practical human post-processing. This research could serve as a good stepping stone to tackle an even harder Scan2BIM problem.

## Acknowledgment

This research is supported by National Science Foundation under grant IIS 1540012. We thank Floored Inc. for providing us with the dataset and support. We thank Eric Turner and Claudio Mura for running their code on our datasets for evaluation.

## References

- [1] <http://pointclouds.org/>.
- [2] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. *Proc. CVPR*, 2014.
- [3] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principle component analysis? *Journal of the ACM*, 58(3):11:1–11:37, 2011.
- [4] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1), 2012.
- [5] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3d primitives for single image understanding. In *International Conference on Computer Vision*, pages 3392–3399. IEEE, 2013.
- [6] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *Computer Vision and Pattern Recognition*, 2009.
- [7] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *International Conference on Computer Vision*, 2009.
- [8] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Computer Vision and Pattern Recognition*, 2010.
- [9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2015.
- [10] M. Golparvar-Fard, F. Peña-Mora, C. A. Arboleda, and S. Lee. Visualization of construction progress monitoring with 4d simulation model overlaid on time-lapsed photographs. *Journal of Computing in Civil Engineering*, 23(6):391–404, 2009.
- [11] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *International Conference on Computer Vision*, 2013.
- [12] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3d scene reconstruction and class segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 97–104. IEEE, 2013.
- [13] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *International Conference on Computer Vision*, 2009.
- [14] E. Herbst, P. Henry, and D. Fox. Toward online 3-d object segmentation and mapping. In *International Conference on Robotics and Automation*, 2014.
- [15] S. Ikehata, H. Yan, and Y. Furukawa. Structured indoor modeling : Project website. <http://www.cse.wustl.edu/~furukawa/sim>.
- [16] Z. Jia, A. Gallagher, A. Saxena, and T. Chen. 3d-based reasoning with blocks, support, and stability. In *Computer Vision and Pattern Recognition*, pages 1–8, 2013.
- [17] Z. Jia, A. Gallagher, A. Saxena, and T. Chen. 3d-based reasoning with blocks, support, and stability. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2013.
- [18] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *Proc. ACM SIGGRAPH*, 2013.
- [19] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):138, 2012.
- [20] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87*, pages 163–169, New York, NY, USA, 1987. ACM.
- [21] C. Mura, O. Mattausch, A. J. Villanueva, E. Gobbetti, and R. Pajarola. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, 2014.
- [22] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012.
- [23] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz. Occluding contours for multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4002–4009. IEEE, 2014.
- [24] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.
- [25] N. Silberman, L. Shapira, R. Gal, and P. Kohli. A contour completion model for augmenting surface reconstructions. In *Computer Vision–ECCV 2014*, pages 488–503. Springer, 2014.
- [26] S. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *International Conference on Computer Vision*, 2009.
- [27] M. Tarini, N. Pietroni, P. Cignoni, D. Panizzo, and E. Puppo. Practical quad mesh simplification. *Proc. of Eurographics*, 2010.
- [28] Trimble. Sketchup. <http://www.sketchup.com>.
- [29] E. Turner, P. Cheng, and A. Zakhor. Fast, automated, scalable generation of textured 3d models of indoor environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):409–421, 2015.
- [30] J. Xiao and Y. Furukawa. Reconstructing the worlds museums. In *European Conference on Computer Vision*, pages 668–681. Springer, 2012.
- [31] X. Xiong, A. Adan, B. Akinci, and D. Huber. Automatic creation of semantically rich 3d building models from laser scanner data. *Automation in Construction*, 31:325–337, 2013.
- [32] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition. *Advances in Neural Information Processing Systems*, 2014.