Reconstructing Building Interiors from Images

Yasutaka Furukawa, Brian Curless, Steven M. Seitz
University of Washington, Seattle, USA
{furukawa,curless,seitz}@cs.washington.edu

Richard Szeliski
Microsoft Research, Redmond, USA
szeliski@microsoft.com

Abstract

This paper proposes a fully automated 3D reconstruction and visualization system for architectural scenes (interiors and exteriors). The reconstruction of indoor environments from photographs is particularly challenging due to texture-poor planar surfaces such as uniformly-painted walls. Our system first uses structure-from-motion, multi-view stereo, and a stereo algorithm specifically designed for Manhattan-world scenes (scenes consisting predominantly of piece-wise planar surfaces with dominant directions) to calibrate the cameras and to recover initial 3D geometry in the form of oriented points and depth maps. Next, the initial geometry is fused into a 3D model with a novel depth-map integration algorithm that, again, makes use of Manhattan-world assumptions and produces simplified 3D models. Finally, the system enables the exploration of reconstructed environments with an interactive, image-based 3D viewer. We demonstrate results on several challenging datasets, including a 3D reconstruction and image-based walk-through of an entire floor of a house*, the first result of this kind from an automated computer vision system.*

1. Introduction

3D reconstruction and visualization of architectural scenes is an increasingly important research problem, with large scale efforts underway to recover models of cities at a global scale (e.g., Google Earth, Virtual Earth). While user-assisted approaches have long proven effective for facade modeling [7, 16, 24, 19], an exciting new development is the emergence of *fully-automated* approaches for the reconstruction of urban outdoor environments both from ground-level and aerial images [5, 17, 26].

Unfortunately, if you walk *inside* your home and take photographs, generating a compelling 3D reconstruction and visualization becomes much more difficult. In contrast to building exteriors, the reconstruction of interiors is complicated by a number of factors. First, interiors are often dominated by painted walls and other **texture-poor** surfaces. Second, **visibility reasoning** is more complicated for interiors, as a floor plan may contain several interconnected rooms, with only a small subset visible in each photo. Third, capturing an entire house interior poses a significant **scalability** challenge, particularly given the prevalence of thin structures such as doors, walls, and tables that demand high resolution relative to the scale of the scene. These factors pose significant problems for multi-view stereo methods (MVS) that perform relatively poorly for interior scenes.

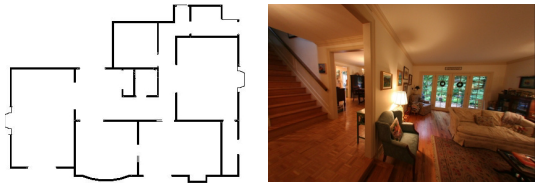

Figure 1: Floor plan and photograph of a house interior.

Our goal is a *fully automatic* system capable of reconstructing and visualizing an entire house interior. The visualization should be compelling and photorealistic. Taking inspiration from image-based rendering approaches [7, 11], we seek to reconstruct *simple* models, reminiscent of the floor plan in Fig. 1, which can then be rendered with the input images to hallucinate visual detail and parallax not actually present in the reconstruction. Our approach builds on a number of existing techniques to achieve this goal. Indeed, our primary contribution is the design of a system that significantly advances the state of the art for automatically reconstructing and visualizing interiors.

We start with the observation that, as illustrated in Fig. 1, architectural scenes often have strong planar arrangements, and their floor plans, though sometimes irregular, are usually highly structured, typically with walls (and floors and ceilings) aligned with one of three orthogonal axes. Accordingly, we invoke the *Manhattan-world* assumption [6], which states that all surfaces are aligned with three dominant directions, typically corresponding to the X, Y, and Z axes. Clearly, restricting some surfaces to these orientations will help recover large-scale structures such as the walls aligned with the floor plan. We take this to an extreme by requiring that *all* reconstructed surfaces have this restriction; even "lumpy" objects will be reconstructed as a union of box-like structures. These structures then serve as

2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

geometric proxies for image-based rendering (IBR).

Given a set of images of a scene, we first use structure-from-motion [20] and multi-view stereo (MVS) [9] software to calibrate the cameras and obtain an initial 3D reconstruction in the form of oriented 3D points. Next, a stereo algorithm designed specifically for Manhattan-world scenes is used to generate axis-aligned depth maps for the input images [8]. Then, the depth maps and MVS points are merged into a final 3D model using a novel depth-map integration algorithm that: (1) poses the reconstruction problem as a volumetric MRF on a binary voxel grid, (2) solves it using graph-cuts, and (3) extracts a simplified, axis-aligned mesh model. Finally, the system provides virtual exploration of reconstructed environments with an interactive, image-based 3D viewer. To our knowledge, our system is the first fully automated computer vision system to reconstruct and enable the walkthrough of an entire floor of a house.

1.1. Related Work

Automated 3D reconstruction algorithms can be roughly classified as either *model-based* approaches that reconstruct scenes composed of simple geometric primitives [5, 7, 16, 24], or *dense* approaches where the objective is to capture fine details without strong priors.

Multi-view stereo (MVS) is one of the most successful dense approaches and produces models whose accuracy rivals laser range scanners [18]. However, MVS requires texture and thus architectural scenes pose a problem due to the prevalence of texture-poor, painted walls. Furthermore, MVS focuses on producing high resolution 3D models, which we argue is overkill for architecture, which consist largely of flat walls. While it is possible to simplify MVS models of architecture as a post-process, we show that this approach yields disappointing results. Model-based approaches, on the other hand, incorporate scene-specific constraints to improve the robustness of reconstruction. Notable examples include Cornelis et al. [5], who reconstructed entire city streets from car-mounted video by restricting the geometry to vertical ruled surfaces.[1] Aerial images are used for reconstructing building models in [26], where the system uses a height field, a rough building mask, and 3D lines to segment out buildings from images and to reconstruct roof shapes. Their system produces very impressive results for outdoor environments, but it relies heavily on obtaining dense, accurate stereo (more problematic for interiors, as we demonstrate).

Our depth map integration approach is similar to several existing MVS approaches [13, 25], where they first encode depth map information into a voxel grid covering the 3D space of interest, perform a volumetric reconstruction on voxels, and use marching cubes [15] to extract a mesh model. However, these techniques produce extremely detailed and complex 3D models; operating on the scale of a full house interior poses a major challenge. We instead focus on extracting very simple 3D models that exploit known properties of architectural scenes. Our 3D viewer uses a reconstructed mesh model as a geometric proxy for view-dependent texture mapping as used in [7, 21] and Façade. Other IBR methods that leverage a geometric proxy, e.g., unstructured lumigraphs [4], could be used instead.

[1] Pollefeys et al. also reconstruct street-side views while exploiting certain structural information, but their system is a dense approach producing high resolution models.

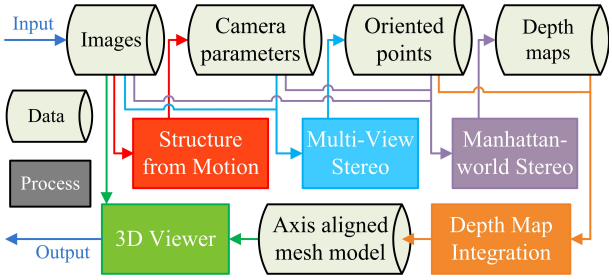

Figure 2: 3D reconstruction and visualization system for architectural scenes. See text for details.

2. System Pipeline

This section describes the overall pipeline of our proposed 3D reconstruction and visualization system illustrated in Fig. 2. Our system consists of four steps.

Camera Calibration and Initial Reconstruction
Given a set of images of a scene, the first step is to compute camera viewpoints and parameters; we use the Bundler Structure from Motion (SfM) package for this purpose [20]. Next, multi-view stereo (MVS) software *PMVS* [9], which is also publicly available, is used to reconstruct dense 3D oriented points, where each oriented point is associated with its 3D location, surface normal, and a set of visible images.

Manhattan-world Stereo
Due to lack of texture and other challenges, MVS produces incomplete models for most architectural scenes. Therefore, some form of interpolation is needed to compute a full model from the oriented points reconstructed in the previous step. For this purpose, our system uses a stereo algorithm proposed by Furukawa et al. [8], which exploits the *Manhattan-world* assumption: surfaces are piece-wise planar and aligned with three dominant directions. The output of this algorithm is a complete depth map for each input image, and the algorithm operates as follows. First, it identifies three dominant orientations in the scene from the surface normal estimates associated with MVS points. Second, it generates a set of candidate planes along each dominant axis on which most of the geometry lies by extracting peaks from the density of MVS points projected onto the

axis. Lastly, it recovers a depth map for an input image by assigning one of the candidate planes to each pixel, which is posed as a Markov random field (MRF) and solved with graph cuts [3]. Refer to [8] for more details.[2]

Axis-aligned Depth Map Integration
Our final reconstruction step is to merge axis-aligned depth maps into a 3D mesh model. Details are given in Section 3.

3D Viewer
Given a 3D model of a scene, the system allows users to explore the reconstructed environment using an interactive image-based 3D viewer. As mentioned in the introduction, a reconstructed model is used as a geometric proxy for view-dependent texture mapping as in [7, 21], where two images are used for alpha-blending in every frame, with the blending weights being inversely proportional to the distances between the current viewpoint and the optical centers of the cameras used for texture-mapping. The viewpoint can be controlled in the following two navigation modes. In the first mode, a user simply navigates through the input images in a pre-specified order (e.g., the order in which the pictures were taken). In the second mode, a user has a restricted free-form 6-DOF navigation: A viewpoint can move freely in 3D space, while the viewer keeps track of distances to all the cameras. Whenever the closest camera changes, the viewpoint automatically moves to that camera with rotational motion to align viewing positions and directions, after which a user has free control over the viewpoint again. In this mode, since we have a full 3D model of an environment, the viewer performs collision detection to avoid, e.g., passing through walls.

3. Axis-aligned Depth Map Integration

Recent MVS algorithms have shown that first reconstructing depth maps and then merging them into a surface can yield competitive reconstructions [18]. We follow this approach, starting from Manhattan-world depth maps [8]. We formulate an objective over a cost volume with binary labels and optimize with graph-cuts. In this section, we describe our formulation, surface extraction algorithm, and several enhancements suitable for architectural scenes.

3.1. Problem Setup and Notation

The input to the volumetric reconstruction step is a set of depths maps $\{D^1, D^2, \cdots\}$ containing axis-aligned surfaces reconstructed by a Manhattan-world stereo algorithm [8], and the associated three dominant axes of the scene.[3] We first compute the smallest axis-aligned bounding box that contains all the depth maps and optical centers

[2]For the *gallery* dataset, which is the most complicated, we also enforce depthmaps to be inside the smallest axis-aligned bounding box that contains MVS points.

[3]Extracted dominant axes may not be perfectly orthogonal to each other. In that case, our voxel grid would be skewed, but the rest of the algorithm is independent of the orthogonality of the axes. Here, we assume that the axes form orthogonal bases for simplicity.

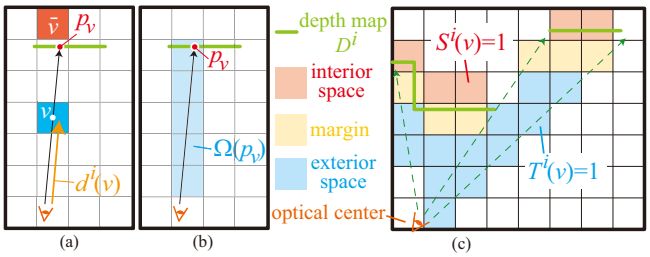

Figure 3: (a) Given a voxel v, p_v denotes the pixel on depth map D^i that is closest to the image projection of v, while $\bar{v}$ is the voxel immediately behind p_v. (b) $\Omega(p_v)$ denotes a set of voxels that is in front of p. (c) Each depth map votes for interior and exterior assignments.

of the input cameras. The resolution of the voxel grid is determined by a user-controlled parameter N_r that specifies the number of voxels along the longest dominant direction.

Given a depth map D^i and a voxel v, p_v is used to denote a pixel on D^i (and the corresponding 3D point on D^i interchangeably) that is closest to the image projection of the voxel center, and $\bar{v}$ denotes the voxel that is immediately behind p_v on the opposite side from the camera (Fig. 3a–b). $\Omega(p_v)$ denotes the set of voxels intersected by the ray from p_v to the optical center of D^i. Finally, $d^i(v)$ is used to represent the depth value of v with respect to D^i.

3.2. Volumetric Reconstruction

Following recent work in MVS surface reconstruction [13, 23], we set each voxel to have a binary label, either *interior* or *exterior* (`int` or `ext`). [4] We use graph-cuts [3] to minimize an objective of the form:

$$F = \sum_{v \in V} F_1(l_v) + \lambda_1 \sum_{v \in V, u \in \mathcal{N}(v)} F_2(l_v, l_u), \qquad (1)$$

where l_v is the label assigned to voxel v, $\mathcal{N}(v)$ denotes neighboring voxels, and λ_1 controls the relative effect of the unary and binary functions, F_1 and F_2, respectively.

In previous formulations, $F_2(l_v, l_u)$ has encoded the observations of the surface location. Evidence (from photo-consistency or extracted depths) that a surface should appear between two voxels can be used to relax the cost of transitioning from interior to exterior. This term implicitly seeks a weighted minimal surface, a kind of regularization that favors constructing tight surfaces, particularly where no data otherwise drives the surface. By contrast, F_1 has either been an inflating term [23]—a small constant penalizing empty voxels, to avoid the degenerate solution of no surface at all—or has been informed by visibility information derived from depth maps [13]. When implementing the

[4]Interior and exterior voxels are also referred to as full and empty, respectively.

graph, a 26-neighborhood around each voxel is preferred to help recover oriented surfaces, although 6-neighborhoods (neighbors along coordinate axes) are often used to reduce memory footprint and accelerate computation [23].

Our formulation departs from these approaches in the following ways. First, to favor simple surfaces, we use the regularizing effect of minimal surface area as our smoothness term by setting $F_2(l_v, l_u) = \delta(l_v = l_u)$. Note that this term does not depend at all on the observations. Instead the observations—depth maps in our case—are used to drive the unary term F_1, providing evidence for the emptiness or fullness of voxels based on visibility.[5] In addition, we consciously employ a 6-neighborhood around voxels and add a *shrinkage* term (see below) to the energy that penalizes full voxels. This simple combination leads to small numbers of clean, sharp corners in areas where no observations drive the surface, which is a desirable property for reconstructing architectural interiors.[6] The minimum area and volume terms might seem to drive toward an empty volume solution. However, our Manhattan-world depth maps are quite dense, providing ample data to avoid such a collapse.

Next we describe our definition of the unary term $F_1(l_v)$. Similar to [13], we base this term on the visibility information implied by the depth maps. We employ an occupancy voting scheme. If the depth maps vote that a voxel should be empty (due to visibility), we assign a high cost to $F_1(l_v = \texttt{int})$. If a voxel is just behind reconstructed depths in many views, it is likely occupied, and a high cost should be assigned to $F_1(l_v = \texttt{ext})$. We can accumulate votes for each voxel by summing over the depth maps as follows: $I(v) = \sum_{i=1}^{N_c} I^i(v), \quad E(v) = \sum_{i=1}^{N_c} E^i(v)$.
$I^i(v)$ and $E^i(v)$ are the amount of evidence for being interior and exterior, respectively, based on a single depth map D^i, and N_c is the number of input cameras. We set $I^i(v)$ to be 1 where v should be interior, that is, immediately behind D^i, and $E^i(v)$ to be 1 where v should be exterior, that is, in front of D^i (see Fig. 3c):

$$I^i(v) = 1 \quad \text{if } 0 < d^i(v) - d^i(p_v) \leq \mu, \tag{2}$$
$$E^i(v) = 1 \quad \text{if } 0 < d^i(v) \text{ and } d^i(v) - d^i(p_v) \leq -\gamma.$$

Note that $I^i(v)$ and $E^i(v)$ are 0 in all the other cases, and γ is a margin to allow errors in depth maps, which is set to twice the voxel resolution ($\gamma = 2\mu$) in our experiments.

We could now set $F_1(l_v = \texttt{int}) = E(v)$ and $F_1(l_v = \texttt{ext}) = I(v)$. In practice, however, we have found that depth maps generated by Manhattan-world stereo can occasionally go quite astray from the true surface. Implicit in our voting scheme (and made explicit in [13]) is the assumption that the certainty (and thus weight) of visibility information provided by a pixel in a depth map depends only on the reconstructed depth at that pixel. However, the depths at some pixels are inconsistent with other depth maps, and should be down-weighted. In particular, we set the label costs to:

$$F_1(l_v = \texttt{int}) = \Sigma_{i=1}^{N_c} \omega^i(p_v)\psi^i(v)E^i(v), \tag{3}$$
$$F_1(l_v = \texttt{ext}) = \Sigma_{i=1}^{N_c} \omega^i(p_v)I^i(v). \tag{4}$$

where $\omega^i(p_v)$ measures the confidence of the depth at pixel p_v based on consistency with other depth maps, and $\psi^i(v)$ is a correction term that balances volumetric costs against surface costs (more on this below). We define $\omega^i(p_v) = \exp\left[\frac{1}{8}I(\bar{v}) - \lambda_2\left(E(\bar{v}) + \sum_{v' \in \Omega(p_v)} I(v')\right)\right]$. The first term $I(\bar{v})$ in the exponent is the number of depth maps that pass through the same location p_v, and hence, agree with its depth. The next terms measure conflicts (and are thus subtracted from the number of agreements). The first conflict term is the number of depth maps whose exterior space, which should be empty, contains p_v. The second conflict term is the number of times the space from p_v to its depth map's center of projection, which again should be empty, contains depths observed by other cameras. Intuitively, the weight $\omega^i(p_v)$ becomes smaller when the measure of agreement $I(\bar{v})$ is small or that of conflict $E(\bar{v})$ and $I(v')$ is large. Values chosen for λ_2 are given in Table. 1.

Finally, $\psi^i(v)$ reduces the effects of exterior information $E^i(v)$ exponentially based on the distance from the depth map to v: $\psi^i(v) = \min\left(1, \exp\left[-\frac{d^i(p_v) - d^i(v) - \gamma}{8\mu}\right]\right)$.
The motivation for this scaling term is that $F_1(l_v = \texttt{int})$ and $F_1(l_v = \texttt{ext})$ as defined in Equation 4, would have different units otherwise: The interior evidence lies immediately behind depth maps, and hence, covers a 2D manifold, while exterior evidence covers a 3D space in front of the depth map. The unit difference is particularly problematic when input depth maps contain large errors.

Minimum-volume solution: We have observed that the global minimum of the energy function (1) is often not unique, because the smoothness term F_2 alone is ambiguous where there is no data information. Figure 4 illustrates an example (in 2D) of a depth map and its corresponding minimum-volume and maximum-volume solutions. As we use a 6-neighborhood in 3D, these solutions have the same amount of energy; i.e., the area is measured in voxel faces, and a monotonic, jagged path has just as many boundary faces as two axis-aligned planes that cleanly form a corner. Indeed, there is a family of surfaces between these two that

[5] Hernández et al. [13] use both a data-dependent binary term, and visibility-driven (thus also data-dependent) unary term. We encode the entire data contribution in the latter.

[6] One could certainly imagine more direct schemes for encouraging simple, axis-aligned reconstructions, e.g., directly computing a final 3D model from MVS points without the Manhattan-world stereo step. Indeed, we experimented with smoothness penalties based on higher-order cliques that prefer axis-aligned piece-wise planar surfaces. However, the energy terms became non-submodular. We tested several approximation algorithms such as quadratic pseudo-boolean optimization (QPBO) [12] and a submodular-supermodular procedure, but the optimizations settled in undesirable local minima. In practice, the simple smoothness penalty F_2 described above produces satisfactory results.

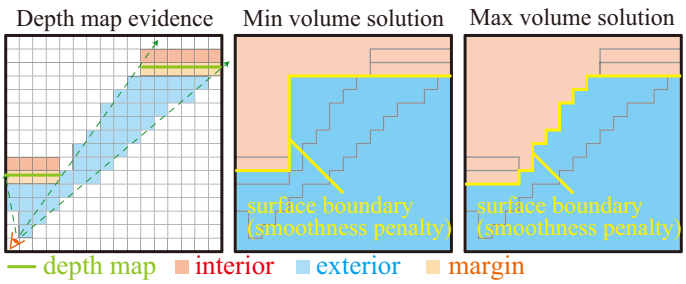

Figure 4: The global minimum of our energy function (1) is usually not unique. We choose the minimum-volume surface as the solution, which tends to be less jagged. Yellow lines in the right two figures represent the amount of smoothness penalties; the total penalties are equal in these two cases.

have the same energy. In general, the minimum-volume surface is smoother, because it can cut, without penalty along straight paths into regions that are not observed. By contrast, the maximum volume surface tends to be jagged, as it must conform to the boundary (between blue and white in the figure) constrained by visibility. Surfaces between the minimum and maximum will also be more jagged than the minimum solution. In practice, we add a small penalty (10^{-6}) to $F_1(l_v = \texttt{int})$, and find the solution with the minimum volume. It is straightforward to show that these same arguments do *not* hold for 26-neighborhoods, where the minimum volume solution will not in general result in two planes meeting at a corner in cases such as the one described here.

3.3. Mesh Extraction with Sub-voxel Refinement

After using graph cuts [3] to find the optimal label assignment, we triangulate the boundary between interior and exterior voxels to extract a mesh model, where the boundary is defined as a set of voxel faces whose incident voxels have different labels. We also refine vertex positions up to sub-voxel accuracy while keeping the axis-aligned structure. More concretely, for each (axis-aligned) slice of the voxel grid, we identify a set of voxel faces that belong to the surface boundary. For every connected component C, we then create a constrained delaunay triangulation [2], where the boundaries of the component (including any holes) are constrained to be part of the triangulation (see Fig. 5). After repeating this for all slices, the result is a sparsely triangulated mesh with faces and edges aligned with the voxel grid. Next, we compute a sub-voxel offset along the normal direction to each component C by: (1) collecting MVS points that are within 1.5μ of C and have a compatible normal (i.e., within 45 degrees of C's normal), and (2) computing the (signed) distance to the centroid of the collected MVS points. The vertices of C are then shifted by this distance in C's normal direction. We repeat this process for every connected component, visiting each exactly once.

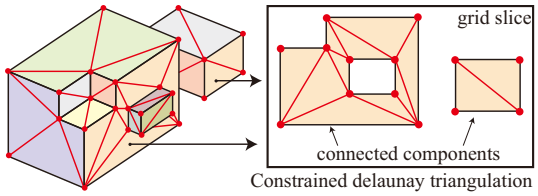

Figure 5: Delaunay triangulation is used for each connected component on each grid slice to triangulate the reconstructed volume.

3.4. Enhancements

We now describe several optimizations and implementation details suited to architectural reconstruction.

Grid pruning: When reconstructing large-scale scenes, we adapt the voxel resolution spatially according to the input depth maps. This adaptation accelerates the reconstruction, reduce memory footprint, and provide a level-of-detail control. Octree partitioning of space is one possible adaptation strategy [13]. However, this approach does not take advantage of the fact that floor plans often have large, unevenly spaced planes, some of which align with multiple disconnected walls. Instead, we take a simple pruning approach, removing voxel slices that are not supported by the depth maps. In particular, given a pixel in a depth map, we define it to be a *grid* pixel if it corresponds to a corner or an edge of the depth map, i.e., if its candidate plane assigned by the Manhattan-world stereo algorithm is different from that of a neighboring pixel. Intuitively, grid pixels occur most often at scene corners and edges, and hence, we want grid slices to pass through them. So, for each grid slice, we count the number of grid pixels (their 3D locations) that are within $\mu/2$ distance of the slice and prune the slice away if the count is less than threshold λ_3. We modify the problem formulation to account for the larger voxel cells, which implicitly contain the original fine resolution cells, using a procedure known as *contraction* [12]. See Tables 1 and 2 for the values of λ_3 and the amount of speed-up achieved by this procedure. Fig. 6 shows how changing the value of λ_3 can be used to control model complexity; raising the threshold removes small scale structure to create simpler models.

Ground plane determination: Ground planes may not be clearly visible in the input images (e.g., when photos are taken by pointing the camera horizontally), and hence, are not reconstructed well in the depth maps. In these cases, Manhattan-world stereo may extrapolate surfaces below the ground plane, thus increasing the bounding box of the volume and reconstructing surfaces below ground. We tighten the bounding volume to match the ground plane as follows. Among the six (directed) dominant directions, the upwards vertical axis is determined by the one that is compatible (angle difference is less than 45 degrees) with the most num-

	kitchen	hall	house	gallery
# images (N_c)	22	97	148	492
image res [pixel]	3M	3M	2M	2M
# voxels (N_r)	128	150	256	512
# faces	1364	3344	8196	8302
voxel num ratio	0.15	0.036	0.071	0.0052
λ_1	1	1	1	1
λ_2	1/16	1/16	1/16	1/16
λ_3	100	100	10000	15000
run time (SfM)	13	76	92	716
run time (MVS)	38	158	147	130
run time (MWS)	39.6	281.3	843.6	5677.4
run time (DI)	0.4	0.4	3.6	22.4

Table 1: Characteristics of datasets. See text for details.

ber of camera up-vectors. Then, for each horizontal volume slice (of the original volume), we count the number of associated grid pixels and compute the average of the non-zero counts. The ground plane is determined to be the bottommost slice with a count greater than this average, and the bounding volume is then restricted to go no lower than this ground plane. (For interiors, a ceiling plane could be needed as well, though this proved unnecessary in our examples.)

Boundary filling: After the volumetric label optimization but before the surface extraction step, we mark all the voxels at the boundary of the voxel grid to be interior. Since the voxel grid contains the optical centers of the input cameras, this boundary filling creates a surface, to which an image texture can be mapped and rendered in the 3D viewer. This is particularly useful for the visualization of areas that are not reconstructed, such as distant scenery. (See our project website [1] for examples.)

4. Results and Conclusion

We used the Bundler [20] SfM package for camera calibration, and the PMVS [9] MVS package for initial geometry reconstruction; both packages are publicly available. The rest of our system is implemented in C++, run on a dual quad-core 2.66GHz PC. Four datasets are used in our experiments whose sample images are shown in Figures 1 and 6, and summarized in Table 1. From the top, Table 1 lists the number of input images, (approximate) image resolution in pixels, the number of voxels along the longest dimension, and the number of faces in the final mesh model. The next row is the ratio of the total number of voxels in the simplified grid to that in the original uniform grid. Notice that the ratio is as small as 0.5 to 7 percent for *hall*, *house*, and *gallery*. λ_1, λ_2, and λ_3 are the parameters in the depth map integration step. A large value is set to λ_3 for *house* and *gallery*, so that the final 3D models would be simple.

The last four rows of the table provide the running time of SfM, MVS, Manhattan-world stereo (MWS), and the depth map integration (DI) steps in minutes. Note that the MWS step is currently the bottleneck of the system, where it computes depth maps for all the input images. To accelerate this step, each depth map could be computed in parallel (a few minutes per depth map).

Figure 6a shows intermediate reconstructions, in particular, MVS points and axis-aligned depth maps for the *house* dataset. The final 3D models are shown in Figure 6bd for *house* and *gallery* (see the project website [1] for more results and better visualization on all the datasets). Note that MVS points are sparse in many places, and depth maps are fairly noisy. Nonetheless, our algorithm has successfully reconstructed extremely simple models while discovering large-scale structure such as ground planes or vertical walls each of which is represented by a single or a few planar surfaces. In Figure 6ce, we generate floor plan images of the *house* and the *gallery* datasets (red) by projecting vertical surfaces of the model to the horizontal plane, where a pixel intensity represents the amount of projected surface. We manually align them with actual floor plans to evaluate the accuracy. They are fairly consistent, though there is a small amount of global warping, probably due to calibration (SfM) errors. It is also worth pointing out that walls reconstructed in the middle of *house* and *gallery* are extremely thin, difficult for general MVS algorithms to recover. Our system is capable of reconstructing small objects in a scene such as a juice bottle and a coffee maker in *kitchen*, and sofas, tables, and cabinets in *house* (again see the project website [1] for more visualization of our results). Also note that many objects in the scenes are not necessarily axis-aligned, but reasonably approximated by the system, which yields compelling IBR experiences.

Figure 6f shows our system's capability to control the complexity of a model using parameter λ_3. Two numbers below each figure are the value of λ_3 (left) and the number of faces in a model (right). Note that this complexity control is different from, for example, using a low-resolution voxel grid to obtain a simple model. Our approach may lose small-scale details due to pruning, but still captures large scale structure accurately up to sub-voxel precision. Next, Figure 6g compares the proposed approach with a state-of-the-art multi-view stereo algorithm in which Poisson surface reconstruction [14] is applied to the oriented points generated by PMVS [9]. Note that PMVS+Poisson reconstructions capture more detail in certain places, but they also contain many errors and are too dense for an interactive visualization. To facilitate comparison, we used a popular mesh simplification algorithm *QSlim* [10] to make the number of faces equal to that of our model. However, this simplification procedure just worsened the MVS models: Our system achieves simplification by making use of global structural information of architectural scenes, while QSlim simply relies on local geometric information. Figure 6h shows the effects of the sub-voxel refinement step. Blue

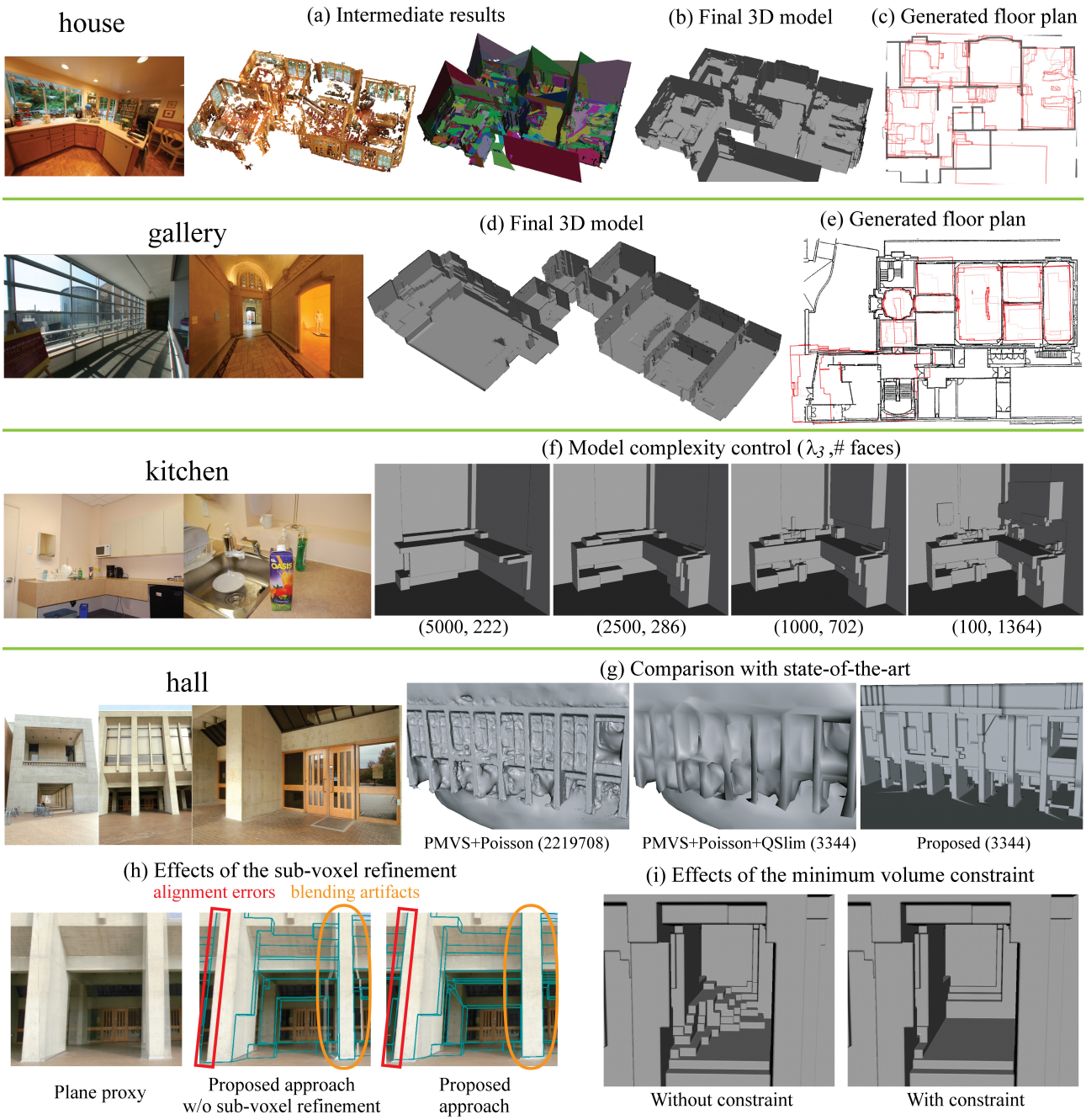

Figure 6: Sample input images are shown for each dataset. (a) Oriented, textured points from PMVS [9] and Manhattan-world depth maps [8] (one color per depth map) for *house*. (b,d) The final 3D models of *house* and *gallery*. (c,e) Generated floor plan images (red) for *house* and *gallery*, and the ground truth (black). (f) Model complexity control with parameter λ_3. (g) Qualitative comparisons with a state-of-the-art MVS approach on *hall* with the number of faces in parentheses. (h) Effects of the sub-voxel refinement procedure. (i) Effects of the minimum volume constraint.

lines in the figure show the wire-frame rendering of the 3D models, with two input images blended together using our IBR viewer. It is easy to see that lines are off from the corresponding image features without the sub-voxel refinement, causing significant blending artifacts, for example, near the wall highlighted with orange ovals. For reference, the left-most figure uses a plane instead of a mesh model to render a scene, which is similar to the Photo Tourism system

Without grid pruning	N_r: # voxels on longest side			
	64	128	256	512
run time (DI) [m]	0.25	1.1	28.7	8764
# voxels	150k	1183k	9400k	74801k
With grid pruning	N_r: # voxels on longest side			
	64	128	256	512
run time (DI) [m]	0.25	0.41	1.6	7.8
# voxels	30k	170k	606k	1097k

Table 2: Effects of the grid pruning on running time.

[21, 22]. As expected, without reasonable geometry, blending artifacts are quite noticeable. Lastly, Figure 6i illustrates the effects of the minimum-volume constraint for the *hall* dataset, which avoids jagged surfaces in regions not reconstructed in depth maps.

Table 2 illustrates the speed-up achieved by the grid pruning as voxel resolution increases. Note that the algorithm does not scale well without grid pruning; the runtime is more than 14 hours without pruning when $N_r = 512$, compared to 8 minutes with pruning. In addition, the number of voxels in the simplified grid (shown in the bottom half of the table) does not increase cubically; growth actually slows as N_r increases, which suggests that the simplified grid captures effective resolution of a scene.

The last step of our system, an interactive IBR system, is demonstrated on the project website [1].

Conclusion We have presented a fully automated system for architectural scene reconstruction and visualization, which is particularly well-suited to challenging textureless scenes such as building interiors. To the best of our knowledge, our system is the first to fully automatically reconstruct and enable the walkthrough of an entire floor of a house. Our future work includes relaxing our axis-aligned surface constraints to handle non-axis aligned structures properly, and to handle even larger-scale scenes such as whole building interiors consisting of multiple floors. Improving our 3D viewer by employing more sophisticated rendering techniques such as Poisson blending is also our future work.

Acknowledgments: This work was supported in part by National Science Foundation grant IIS-0811878, SPAWAR, the Office of Naval Research, the University of Washington Animation Research Labs, and Microsoft. We thank Christian Laforte and Feeling Software for the *kitchen* dataset. We also thank Eric Carson and Henry Art Gallery for the help with the *gallery* dataset.

References

[1] Project website. http://grail.cs.washington.edu/projects/interior.

[2] CGAL, Computational Geometry Algorithms Library.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11), 2001.

[4] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001.

[5] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool. 3d urban scene modeling integrating recognition and reconstruction. *IJCV*, 78(2-3):121–141, July 2008.

[6] J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *ICCV*, pages 941–947, 1999.

[7] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, 1996.

[8] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *CVPR*, 2009.

[9] Y. Furukawa and J. Ponce. PMVS. http://www.cs.washington.edu/homes/furukawa/research/pmvs.

[10] M. Garland. Qslim: Quadric-based simplification algorithm. http://mgarland.org/software/qslim.html.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *Computer Graphics Proceedings*, 1996.

[12] P. L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming*, 28(2):121–155, 1984.

[13] C. Hernández Esteban, G. Vogiatzis, and R. Cipolla. Probabilistic visibility for multi-view stereo. In *CVPR*, 2007.

[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symp. Geom. Proc.*, 2006.

[15] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987.

[16] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. V. Gool. Procedural modeling of buildings. In *SIGGRAPH*, 2006.

[17] M. Pollefeys et al. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2-3):143–167, July 2008.

[18] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 1:519–528, 2006.

[19] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. In *SIGGRAPH Asia*, 2008.

[20] N. Snavely. Bundler: SfM for unordered image collections. http://phototour.cs.washington.edu/bundler/.

[21] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *SIGGRAPH*, 2008.

[22] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*, 2006.

[23] G. Vogiatzis, P. H. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *CVPR*, pages 391–398, 2005.

[24] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based façade modeling. In *SIGGRAPH Asia*, 2008.

[25] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l^1 range image integration. In *ICCV*, 2007.

[26] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *ECCV*, 2008.