Reconstructing the World's Museums

Jianxiong Xiao[1] and Yasutaka Furukawa[2]

[1] Massachusetts Institute of Technology
[2] Google Inc.

Abstract. Photorealistic maps are a useful navigational guide for large indoor environments, such as museums and businesses. However, it is impossible to acquire photographs covering a large indoor environment from aerial viewpoints. This paper presents a 3D reconstruction and visualization system to automatically produce clean and well-regularized texture-mapped 3D models for large indoor scenes, from ground-level photographs and 3D laser points. The key component is a new algorithm called "Inverse CSG" for reconstructing a scene in a Constructive Solid Geometry (CSG) representation consisting of volumetric primitives, which imposes powerful regularization constraints to exploit structural regularities. We also propose several techniques to adjust the 3D model to make it suitable for rendering the 3D maps from aerial viewpoints. The visualization system enables users to easily browse a large scale indoor environment from a bird's-eye view, locate specific room interiors, fly into a place of interest, view immersive ground-level panorama views, and zoom out again, all with seamless 3D transitions. We demonstrate our system on various museums, including the Metropolitan Museum of Art in New York City – one of the largest art galleries in the world.

1 Introduction

3D reconstruction and visualization of architectural scenes is an increasingly important research problem with large-scale efforts underway in a global scale. Google MapsGL seamlessly integrates a variety of outdoor photographic content ranging from satellite, aerial and street-side imagery to community photographs. Indoor environments have also been active targets of photographic data capture with increasing business demands. For example, Google Art Project allows exploration of museums all over the world as well as close examination of hundreds of artworks photographed at high resolution.

However, unlike outdoor environments offering imagery from both aerial and ground-level viewpoints, indoor scene visualization has so far been restricted to ground-level viewpoints, simply because it is impossible to take pictures of indoor environments from aerial viewpoints. The lack of aerial views hampers effective navigation due to the limited visibility from the ground level, especially for large scale environments, such as museums and shopping malls. Without a photorealistic overview, users can easily get lost or confused during navigation.

This paper presents a 3D reconstruction and visualization system for large indoor environments, in particular museums. Our system takes registered ground-level imagery and laser-scanned 3D points as input and automatically produces a 3D model with high-quality texture, under the assumption of piecewise planar structure. Our system enables

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 668–681, 2012.

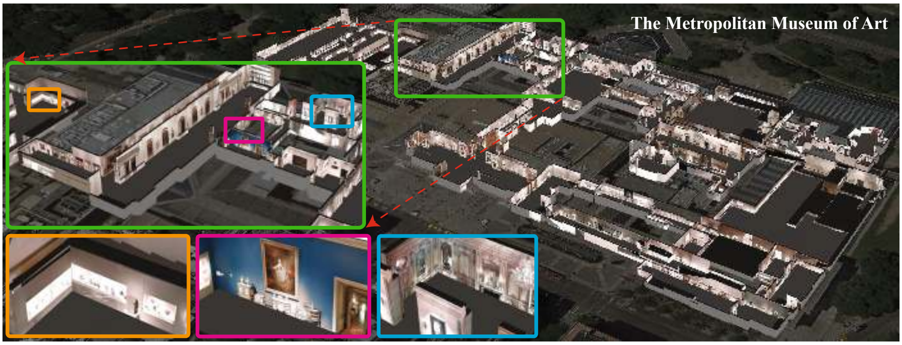

Fig. 1. A texture mapped 3D model of *The Metropolitan Museum of Art* reconstructed by our system, which directly recovers a constructive solid geometry model from 3D laser points.

users to easily browse a museum, locate specific pieces of art, fly into a place of interest, view immersive ground-level panorama views, and zoom out again, all with seamless 3D transitions (demo in the video [1]). The technical contribution is the automated construction of a Constructive Solid Geometry (CSG) model of indoor environments, consisting of volumetric primitives. The proposed "Inverse CSG" algorithm produces compact and regularized 3D models, enabling photorealistic aerial rendering of indoor scenes. Our system is scalable and can cope with severe registration errors in the input data, which is a common problem for large-scale laser scanning (Fig. 2).

1.1 Related Work

Laser range sensors have been popular tools for 3D reconstruction, but the major focus of these approaches has been limited to the reconstruction of small scale indoor environments, objects, and outdoor buildings [2,3]. A human-operated backpack system [4,5] was proposed to produce 3D texture-mapped models automatically, but their results usually have relatively simple structures of the final models. As pointed out in [4], a major challenge for data acquisition is the precise pose estimation for both imagery and 3D sensor data, which is critical for producing clean 3D models with high-quality texture images requiring pixel-level accuracy. Although there exist scalable and precise image-based pose estimation systems based on Structure from Motion algorithms [6], our data collection exposes further challenges. First, indoor scenes are full of textureless walls and require complicated visibility analysis because of narrow doorways. Furthermore, museums often have rooms full of non-diffuse objects (e.g., glass and metallic materials), which violates assumptions of vision-based systems. Active depth sensing usually yields more robust and accurate pose estimation [7,8]. However, even with high-quality laser scans, pose estimation is still very challenging, as museum floors are often uneven (e.g., floors made of stones) causing laser range sensors to vibrate constantly. Furthermore, simply the scale of the problem is unprecedented in our experiments (more than 40,000 images and a few hundred million 3D points for the largest collection), where robustness to registration errors becomes an inevitable challenge to overcome.

In image-based 3D modeling, surface reconstruction is usually formulated as a volumetric segmentation problem via Markov Random Field (MRF) defined over a voxel

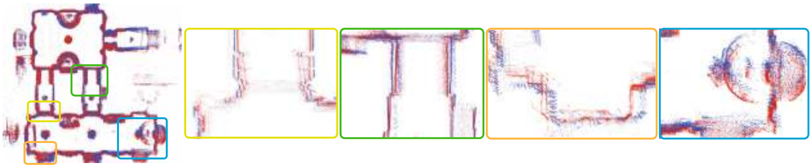

Fig. 2. A top-down view of input laser points, where two different colors represent two vertical laser range sensors. A challenge is the presence of severe noise in the input laser points. Note that errors are highly structured and “double walls” or even “triple walls” are prevalent.

grid [9,10]. However, an MRF imposes only local regularization over pairs of neighboring voxels, which are susceptible to highly structured noise in laser points. Also, these approaches typically penalize the surface area which makes it difficult to reconstruct thin structures such as walls, leading to missing structures or reconstruction holes [9]. The requirement of having a voxel grid limits the scalability of these approaches due to memory limitation. Another popular approach is to impose strong regularity [11,12] by fitting simple geometric primitives to the point cloud data. However, most approaches have focused on modeling outdoor buildings where the structures are much simpler than indoor scenes. Due to these challenges, large-scale indoor scenes are usually modeled by hand in CAD software. An alternative option is to extrude walls from floorplans (i.e. extend the 2D floor plan vertically). However, it is not feasible in practice for our targeted museums, as accurate floor plans do not exist for many museums. Even where they exist, floorplans may come in different styles or formats, which makes it difficult to automatically parse such images. Even if walls are extracted, they need to be aligned with images for texture mapping, which involves a challenging image-to-model matching problem at a massive scale.

For indoor scene visualization, view-dependent texture mapping is typically used to provide interactive scene exploration [9,13]. For example, Google Art Project, Google Maps, and Bing Maps provide immersive panorama-based viewing experiences for museums and indoor businesses. However, unlike in outdoor maps, in all these systems, navigation is restricted to ground-level viewpoints, where photographs can be directly acquired. As a result, there is no effective scene navigation due to the lack of more global overview from aerial viewpoints. Our goal in this work is to produce 3D models that enable aerial rendering of indoor environments using ground-level imagery.

1.2 Data Collection and Preprocessing

A hand trolley is used to capture input data in our system. It is equipped with a rig of cameras and three linear laser range sensors, two of which scan vertically at the left and the right, while the other sensor scans horizontally. For operational convenience, data collection for a large museum is performed over multiple sessions. We use the term *run* to refer to the data collected from a single session. We use the horizontal laser scans to estimate sensor poses for all runs together using [7], and we handle each floor separately. We use the vertical ones for 3D model reconstruction, since they provide good coverage in general. We also estimate a surface normal at each laser point by fitting a local plane to nearby points.

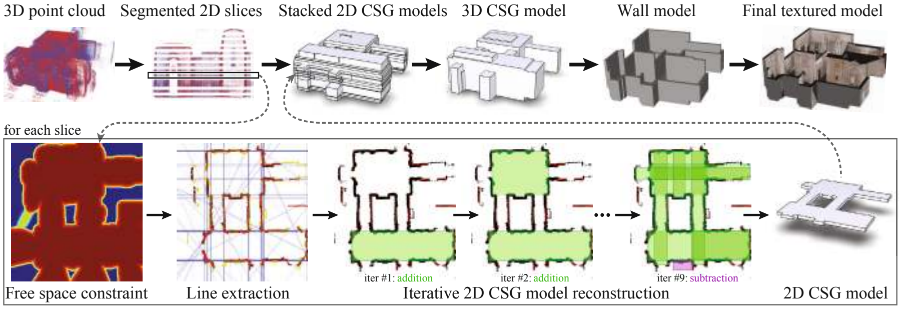

Fig. 3. System Pipeline. The first row shows the entire pipeline of our system, and the second row shows the detailed pipeline of "Inverse CSG" reconstruction of a 2D horizontal slice.

2 Inverse CSG for Large-Scale Indoor Scene Modeling

While many 3D reconstruction algorithms are designed to recover a surface model, we aim at reconstructing a volumetric representation of a scene from registered laser scan. In particular, we model the visible *free space* (surface exterior space), which is directly measured by laser range sensors, as opposed to the invisible *filled-in space*, which is blocked by walls, floors and ceilings. For simplicity, we assume that the space outside buildings is filled-in, which makes the free-space a well-bounded volume.

We reconstruct the free space volume as a Constructive Solid Geometry (CSG) model, which is an expression of simple geometric primitives with union and difference operations. We choose cuboids as volumetric primitives, as they are the most common shapes in architecture design, and good approximations for others. We restrict cuboids to be aligned with the vertical (gravity) direction but allow arbitrary horizontal orientations, which is more general than the Manhattan-world assumption [9]. The use of volumetric primitives and the CSG representation allows us to impose powerful architectural regularization and recover compact 3D models, which is a key factor for large-scale reconstruction and visualization. Our strategy is to enumerate primitive candidates, then construct a CSG model out of the generated primitives to best describe the input laser information. Instead of directly solving for a CSG model with 3D volumetric primitives, where the number of primitive candidates becomes prohibitively large, we (1) split the 3D space into a set of horizontal slices, each of which shares the same horizontal structure (i.e., a floor plan structure in the slice); (2) solve a 2D CSG model with 2D rectangles as geometric primitives in each slice; (3) generate 3D primitives based on the 2D reconstructions, then solve for a 3D CSG model.

2.1 Slice Extraction and 2D Primitive Generation

A floor plan structure on a horizontal slice changes at the existence of horizontal surfaces. For example, in a room with two different ceiling heights, the structure of the

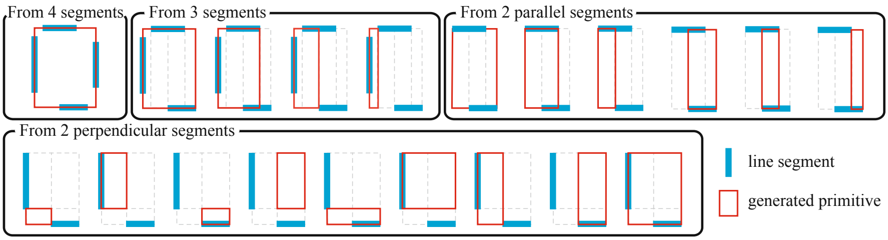

Fig. 4. Line segments are used to form rectangle primitives. See text for details.

horizontal floor plan changes at the two ceilings. We compute the histogram of the number of 3D points in the gravity direction, and convolve it with a Gaussian smoothing operator with a standard deviation equal to 0.5m. We identify peaks in the histogram to identify dominant horizontal surfaces, which divides the 3D space into 2D slices.

Illustrated in the bottom row of Fig. 3, for each slice, we project laser points within the slice onto a 2D horizontal plane, and extract line segments passing through them by Hough transformation. These line segments are used to enumerate rectangle candidates in several ways (Fig. 4). First, a rectangle is generated from every tuple of four line segments forming appropriate angles, where $5°$ error is allowed (also for the remaining cases). Second, every triple of segments forming a "⊏" shape generates four rectangles, where the position of the missing edge is at one of the four endpoints of the two side line segments. Third, every pair of parallel segments generates $\binom{4}{2} = 6$ rectangles, where the positions of the two missing edges are chosen out of the four endpoints of the two line segments. Lastly, every pair of perpendicular line segments is used to generate $\binom{3}{2} \times \binom{3}{2} = 9$ rectangles, where two horizontal (resp. vertical) positions are determined out of the three possible positions (two endpoints of the parallel line segment and the intersection of the two line segments).

We generate an over-complete set of primitives, because line segment extraction may not be perfect, and we do not want to miss any important rectangles. In order to speed up the following reconstruction step, we prune out primitive candidates that are unlikely to be part of the final model, based on the following three criteria. First, any small rectangle, whose width or height is less than 0.15 meters, is removed. Second, identical rectangles are removed except for one copy, where two rectangles are defined to be identical if the distance between their centers is less than 0.3 meters, and the difference of each dimension (width and height) is less than 0.2 meters. Third, a rectangle without enough *supporting* laser points is removed, that is, if the number of laser points that are within 0.2 meters from the boundary of the rectangle is less than 0.05 times the number of laser points in the slice. For a typical run, the algorithm extracts about 50 to 100 line segments per slice, and generates nearly a million primitives initially, which are reduced to a few hundred thousand after pruning.

2.2 Reconstructing 2D CSG Models

We aim to construct a CSG model T that best describes the laser information. The solution space is exponential in the number of hypotheses, and we propose a simple

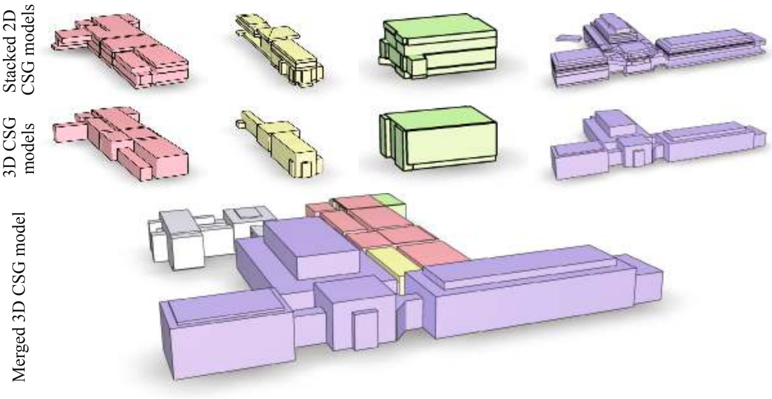

Fig. 5. Reconstruction results for The Frick Collection. Top: Stacked 2D CSG models for individual runs. Middle: Corresponding 3D CSG models. Bottom: The final merged 3D CSG model. The white model is the example illustrated in Fig. 3. Note that there exist three more runs for this museum, which are not visible here because of overlapping or occlusion in rendering.

algorithm that greedily adds or subtracts a primitive. Let $E(T)$ denote an objective function that measures how well T explains the laser information. We start from an empty model. In each iteration, we try to add or subtract each primitive to or from the existing model, evaluate $E(T)$ for each result, and choose the best one to be the new model. The algorithm iterates until $E(T)$ does not increase by more than a threshold ϵ, which is set to 0.02. We now give the definition of $E(T)$ in the rest of this section.

The laser scan provides not only a 3D point cloud, but also the information that nothing exists between the laser center and the scanned 3D point. This *free-space score* is calculated on a grid of voxels in 3D. First, we compute the axis-aligned bounding box of all the laser points, while ignoring the extremal 2% of points in each direction to avoid noise. The voxel size is set to 0.4 meters[1]. For each voxel, we count the number of laser lines passing through it (i.e., a line segment connecting the laser center and the scanned 3D point). A voxel with more counts is more likely to be in the free space, and hence, the free-space score is set to be the count for non-zero count voxels. We truncate the score to be no more than fifteen to avoid bias towards the voxels near the laser centers. Zero-count voxels, in turn, are unlikely to be in the free-space, because laser range sensors usually scan an entire free-space (the trolley operators are instructed to do so). Therefore, we assign a negative free-space score for zero-count voxels. More precisely, for each voxel with zero count, a distance transform algorithm is used to compute the distance to the closest positive-count voxel, and its negated distance (in voxel units) multiplied by 30 is set as the free-space score. The use of distance transform is important as free-space scores tend to become noisy at surface boundaries. To obtain free-space scores for each 2D slice, we simply take the average score value inside the slice vertically.

[1] Different from standard volumetric reconstruction algorithms [9,10], voxel resolution is not critical for accuracy in our approach, as precise surface positions are determined by primitives.

The objective function $E(T)$ consists of the following three terms. The first term measures how much free space information is explained by T:

$$E_1(T) = \frac{\{\text{Sum of free-space scores inside } T\}}{\{\text{Total sum in the domain without negative scores}\}}.$$

Ideally, if the shape T explains the free space perfectly, it would cover all positive scores and none of the negative ones, then $E_1(T) = 1$. The second term measures the ratio of laser points that are explained by T:

$$E_2(T) = \frac{\{\text{\# of points on the surface of } T\}}{\{\text{total \# of points}\}}.$$

However, this term encourages a complex model that explains all the 3D points. Therefore, the third term measures the quality of T from laser points for regularization:

$$E_3(T) = \frac{\{\text{perimeter of } T \text{ near laser points (within 0.2 meters)}\}}{\{\text{total perimeter of } T\}}.$$

The overall objective function $E(T)$ is defined to be a weighted sum of the three terms:

$$E(T) = w_1 W_1(T) + w_2 E_2(T) + w_3 E_3(T), \tag{1}$$

where $w_1 = 1$, $w_2 = 0.1$ and $w_3 = 0.4$.

2.3 Reconstructing 3D CSG Model

Having reconstructed a 2D CSG model for each slice, one possible way to generate a 3D CSG model is to independently extend each 2D primitive to 3D. However, this solution tends to create jagged, misaligned models (e.g. the first row of Fig. 5), as it does not integrate evidence across multiple slices and thus is sensitive to laser scan errors. Instead, we propose a 3D Inverse CSG algorithm to generate a 3D model (e.g. the second row of Fig. 5), using the 2D CSG result to generate 3D candidate primitives. For each 2D primitive in the 2D CSG result, we generate multiple candidate 3D primitives: the vertical positions of their top and bottom faces are chosen from every pair of slice boundary positions (the top and bottom faces have the same shape as the 2D primitive – only their vertical positions vary). Then, we define the same objective function as Eq. 1, which is naturally extended to 3D, and use the same greedy algorithm as in Sec 2.2 to redo everything with 3D cuboids as primitives. After reconstructing a 3D CSG model for each run[2], we simply take the union of all the models in the CSG representation to create the final merged model, and convert it into a mesh model by CGAL [14].

3 Indoor Scene Visualization

Our goal in visualization is to display an entire museum from aerial viewpoints for effective navigation. However, a CSG model reconstructed is not yet suitable for the task, because ceilings and walls occlude interiors, and the walls have no thickness (i.e., a paper-thin model), which look unrealistic from an aerial view. We propose to manipulate the CSG to remove ceilings, add thickness to walls, and lower back-facing walls.

[2] Runs are processed independently for computational and memory efficiency. If needed, a long run can be split into shorter ones, where resulting CSG models can be merged in constant time.

3.1 Wall Model Generation

Denote a reconstructed CSG model from previous section as T, which consists of additive and subtractive primitives (cuboids). We expand T to generate an inflated model T^{fat}, then subtract the volume of T from T^{fat}, which carves out interior volume and leaves a thickened wall structure outside (See Fig. 6). In detail, let Δ be a manually specified thickness of the wall ($\Delta = 0.5$m). We inflate (resp. shrink) each additive (resp. subtractive) primitive horizontally, to move outwards (resp. inwards) each vertical face by Δ. The model is then translated downwards by Δ, so that ceilings are removed when subtracting T. This modified CSG model is denoted as T^{fat}, and $(T^{\text{fat}} - T)$ produces a model with thickened walls and without ceilings. We also limit the maximum height h of a structure by further subtracting $\text{Rect}(h)$: $((T^{\text{fat}} - T) - \text{Rect}(h))$, where $\text{Rect}(h)$ denotes a cuboid whose bottom side is at height h and extends to infinity at the other five directions. h is set to 6 meters. Illustrated in Fig. 6, this construction produces a solid geometry for every vertical facade in a model. However, there may still be some unwanted fragments in the air near complex ceiling structures. Therefore, we construct $((T^{\text{fat}} - T^{\text{up}}) - \text{Rect}(h))$ instead as our wall model. The difference is to construct T^{up} from T by extending the top face of each additive primitive to infinite height, and subtract T^{up} instead of T. We also design a CSG manipulation method to lower back-facing walls for view-dependent optimization. The details of the corresponding CSG operations are given in [1], and example results are shown in Figures 8 and 9.

3.2 Texture Mapping

The last step of our system pipeline is to texture-map a wall model from registered photographs. Modern techniques minimize texture stitching artifacts by processing multiple faces simultaneously [11,15]. However, our system needs to be scalable to an entire

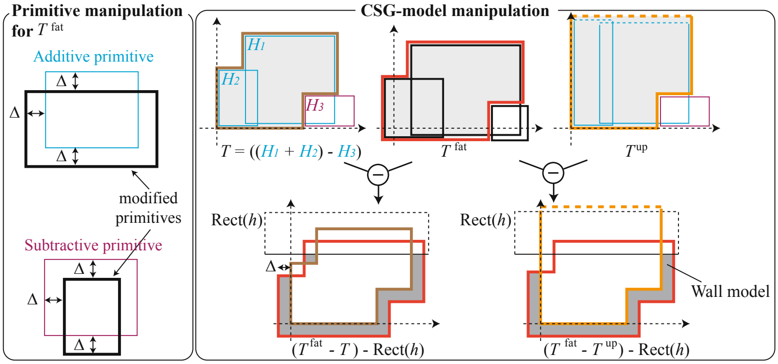

Fig. 6. Given a CSG model T, we expand each additive primitive and shrink each subtractive primitive (left), to generate an inflated model T^{fat}. Subtracting T from this inflated model T^{fat} gives us a wall model with thickness (right). To remove floating walls due to a complex ceiling structure, we subtract T^{up} that is constructed to have infinite height, instead of T.

Table 1. Statistics on input and output data, as well as running time

			National Gallery	Frick	Tretyakov	Hermitage	Uffizi	Met
input	# of runs		1	8	19	20	48	32
	# of images		2,670	8,466	3,274	21,174	11,622	42,474
	# of laser points [million]		12.3	39.0	15.2	75.7	43.8	201.8
output	# of layers (run-average)		9	7.9	5	8.8	7.3	9.8
	# of primitives	additive	14	66	73	137	302	330
		subtractive	2	9	6	23	60	38
	# of triangles		1,050	4,962	3,830	6,902	18,882	22,610
	# of face groups		108	342	282	485	1,123	2,123
run time	sequential [hour]	2D CSG	3.2	4.8	1.9	6.1	19.2	20.1
		3D CSG	4.4	3.2	0.4	13.8	24.4	25.2
	parallel [hour]	2D CSG	3.2	1.0	0.2	0.7	3.9	3.9
		3D CSG	4.4	2.5	0.07	3.0	4.2	2.9

building with tens of thousands of input images, and also be robust against large registration errors among cameras and a 3D model, which is a common problem for laser scanning of a large-scale environment. Our approach is to extract piecewise planar structures in a model as *face groups*, which ideally correspond to walls, floors, and ceilings, then solve a stitching problem in each face group independently[3]. In addition to being scalable, this approach effectively handles pose errors by restricting stitching artifacts at the boundaries of planar structures, such as room corners, where textures are usually not very important. While our CSG representation inherently contains such face group information, the library we used in our implementation – CGAL [14] – does not keep this information when triangulating a mesh model. Also, a single wall is not reconstructed as planar at times, mainly due to the noise in the input laser scan. Therefore, we design a simple region-growing algorithm to identify face groups from a triangulated mesh model: We first pick a face f with the largest area from the mesh to create a new group, and keep adding a neighboring face f' to the group if f and f' are nearly coplanar, until no more faces can be added. Then, we remove all the faces of the group from the mesh and start again from the beginning. For each face group, we use [15] to stitch a texture image with blending. At times, even a single face group becomes large, yielding expensive graph-cuts optimization. Therefore, we solve a stitching problem at a coarse level (roughly one node per $20 \times 20\text{cm}^2$), then upsample the resulting stitching information to generate higher resolution texture images (one pixel per 1cm^2). Lastly, we observed in our experiments that floor textures suffer from very strong artifacts and become mere distractions, mainly because all the cameras are positioned either parallel to the ground or upwards. We identify floor by taking face groups whose vertices are all within 0.05m from the ground, and fill in with a solid gray color. We also assign a gray color to face groups that are not visible from any input camera.

[3] We did consider a large scale graph-cut algorithm [16], but it is still expensive. Our scheme exploits our compact 3D model, and allows easy implementation that works well in practice.

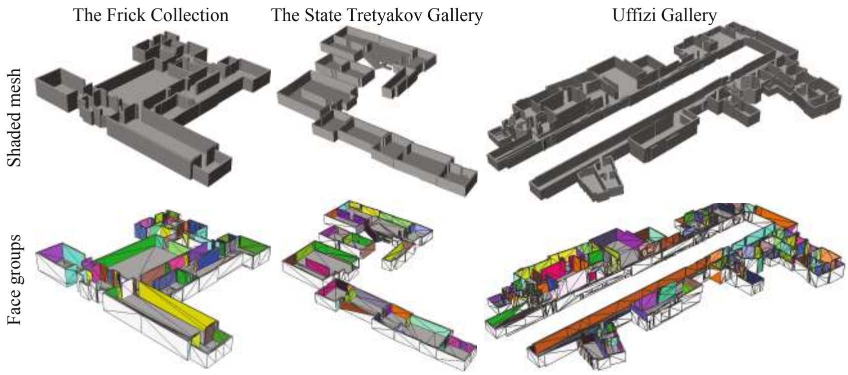

Fig. 7. Shaded renderings of the view-dependent models and the corresponding face-groups. A random color is assigned to each group except for the white faces that are not visible from any input camera and do not require texture stitching, and gray faces that are identified as floors.

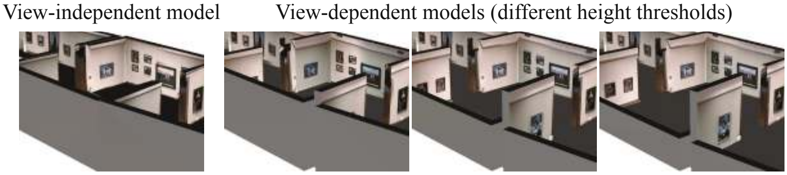

Fig. 8. View-dependent model construction can lower back-facing walls for increased visibility. Different height thresholds are used to generate models on the right.

4 Results and Discussions

We have evaluated our system on a variety of museums in the world. Our smallest data set is *National Gallery* in London consisting of a single run and 2670 images, while the largest one is *The Metropolitan Museum of Art* (Met) in New York City with 32 runs, 42,474 images, and more than two hundred million laser points (See Table 1). The major computation time is on the 3D reconstruction, where the running times for the 2D CSG and the 3D CSG modeling are listed in Table 1. Since computation over multiple runs can be executed in parallel, we list running time for both the serial and the parallel executions. The parallel execution is simulated with a single process, with the maximum running time over all the processes recorded.

Fig. 5 shows reconstruction results of *The Frick Collection*, illustrating the stacked 2D CSG models at the top row, which are further simplified by the 3D CSG model construction. As listed in Table 1, the entire museum is represented by only 75 volumetric primitives (66 additive and 9 subtractive), where the merged mesh model only contains $4,962$ triangles. The figure shows that there is a fair amount of overlap between runs, which has been successfully merged by simply taking the union of their CSG models. A door is usually reconstructed by a box covering the pathway between two rooms (e.g. the second row of Fig. 3).

Shaded renderings of the view-independent wall models and the extracted face groups are shown in Fig. 7, which illustrates that the algorithm successfully identifies major floor and wall structures as face groups. Figures 1 and 9 show final texture-mapped models with some close-ups. The stitched textures are of high quality with very few artifacts, which shows the effectiveness of our face-group identification and texture mapping algorithm. The models in these two figures are obtained by the view-dependent CSG construction for better visibility. The effects of the view-dependent construction are illustrated in Fig. 8, where different thresholds are used to control the height of back-facing walls. Paintings are completely occluded with the view-independent model, but become visible when back facing walls are lowered.

Unlike traditional surface reconstruction from laser scan, it is not our goal to make models as physically accurate as possible, but to reconstruct a global, compact, and well-regularized mesh for high-quality aerial view visualization. It is a totally different challenge, where it makes more sense to compare with vision algorithms designed to handle noise using strong regularization. In Fig. 10, we compare our algorithm with two state-of-the-art techniques – Hernández et al. [10] and Furukawa et al. [9], both of which can usually tolerate noise better than [2]. However, because neither approach is scalable due to the need of a high resolution voxel grid, the smallest dataset *National gallery* is used for comparison. Since both algorithms merge depth maps into a mesh model, each laser point is treated as a single depth map pixel. The output of Hernández's algorithm [10] is a very dense model, where face-grouping and texture mapping fail completely due to pose errors. Therefore, we use a mesh simplification software QSlim [17] to reduce the number of triangles to 2000, which is still twice as many as our 3D model. After reconstruction, we clip their models above a certain height to remove ceilings, then apply our face-grouping and the texture-mapping algorithm. Noisy surfaces and severe stitching artifacts are noticeable for both approaches in Fig. 10, where significant noise in the laser points cannot be regulated well by their Markov Random Field (MRF) formulation. At macro scale, several rooms suffer from space distortions due to the pose error, which breaks the Manhattan-world assumption for [9]. Our algorithm, in turn, succeeds in producing clean texture mapped 3D models, illustrating the effectiveness of the texture mapping algorithm leveraging our highly regularized 3D models. Although Furukawa et al. produces more rooms, a careful look at the textured models would reveal severe artifacts in texture for the extra rooms, because the data capturing device did not enter those rooms. They are partially scanned by cameras and lasers seeing through windows/doorways. Existing methods only enforce local weak regularization and cannot suppress reconstructions of such rooms. We enforce global strong regularization to avoid these rooms where the laser scanner does not reach well.

The lack of ground-truth data prevents us from conducting quantitative evaluations. To qualitatively assess reconstruction accuracy, in Fig. 11, we manual overlay our wall model onto a floor plan image of The Frick Collection. Our model aligns fairly well, which enables interesting hybrid visualization at the right of Fig. 11: The texture-mapped model is rendered on top of the floor plan image, without face-groups identified as floors. Our system is not free from errors as in any other large-scale systems. Small structures such as doorways tend to be missed (e.g., National Gallery in Fig. 9), mostly due to line detection errors. Although our system handles non-Manhattan structures, it is still limited to rectangular structures, and cannot properly model the oval room in The Frick Collection (Fig. 11), or the hexagonal shaped room in National Gallery (Fig. 9).

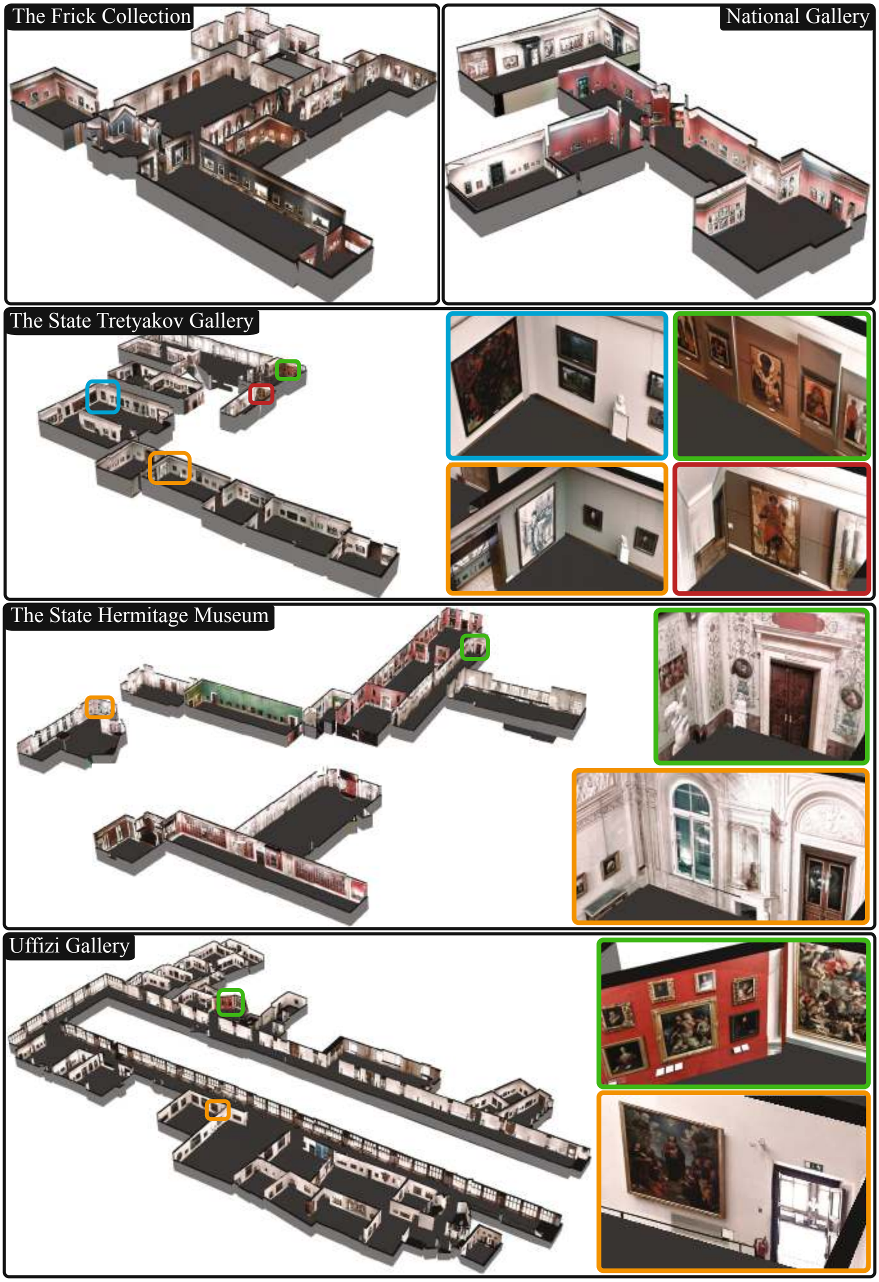

Fig. 9. Final view-dependent texture-mapped 3D models with some close-ups. More results are available in the project website [1].

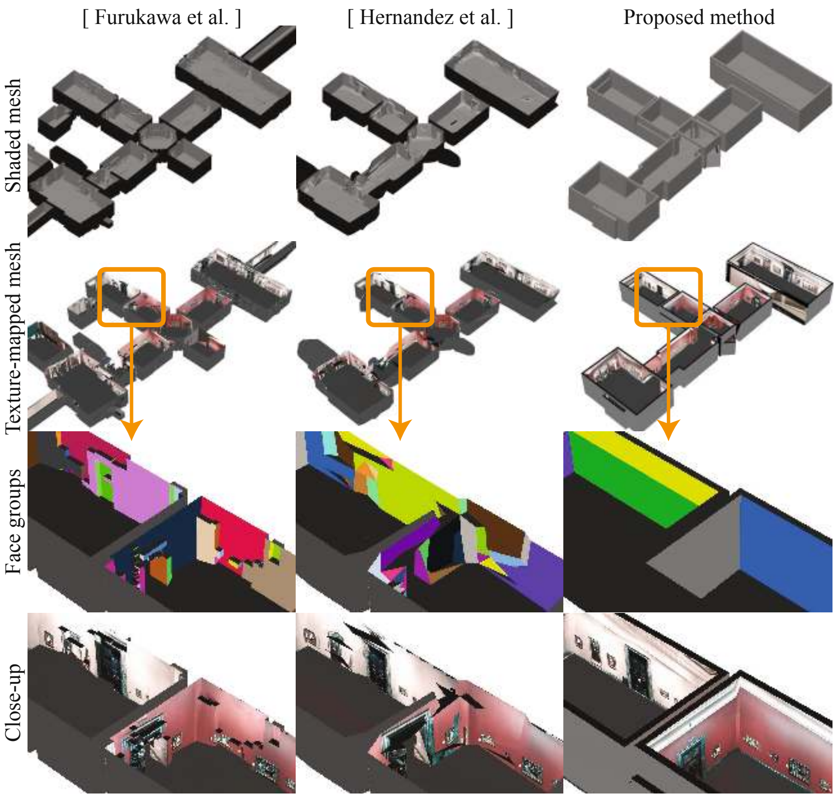

Fig. 10. Comparison with two state-of-the-art 3D reconstruction algorithms [9,10]

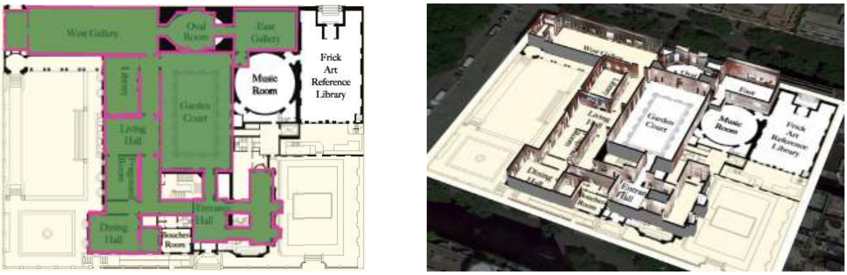

Fig. 11. Left: Manually overlaying our model with a floor plan image of The Frick Collection. For our model, floor is colored in green and walls are in pink. Right: Hybrid visualization where the texture-mapped model is rendered over the floor plan image and a (darkened) satellite imagery

5 Conclusion and Future Works

We have presented a novel indoor modeling and visualization system that directly solves for a CSG model from laser points. Our system has produced high-quality texture-mapped 3D models that are effective for rendering from aerial viewpoints, which we believe opens up new possibilities for indoor scene navigation and mapping. Our future work includes extension of geometric primitive types to more shapes, such as cylinders or spheres. It is also interesting to model objects in a scene, such as sculptures, glass cabinets and tables that are currently missing in our models. Another challenge would be to identify *objects of interests* (e.g. paintings), and associate semantic information to enrich the model. More results and demo can be found in our project website [1].

Acknowledgements. We would like to acknowledge Google Art Project for providing access to the Museums featured. We thank Google Seattle Lightfield team for helpful discussion and support during this work, especially Carlos Hernández, David Gallup, and Steve Seitz. We also thank Maneesh Agrawala and Andrew Owens for brainstorming and discussion. This work was done when Jianxiong Xiao interned in Google. Jianxiong Xiao is supported by Google U.S./Canada Ph.D. Fellowship in Computer Vision.

References

1. Project website (2012), `http://mit.edu/jxiao/museum`
2. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH, pp. 303–312 (1996)
3. Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: Consistently fitting primitives by discovering global relations. In: SIGGRAPH (2011)
4. Liu, T., Carlberg, M., Chen, G., Chen, J., Kua, J., Zakhor, A.: Indoor localization and visualization using a human-operated backpack system. In: IPIN (2010)
5. Sanchez, V., Zakhor, A.: Planar 3d modeling of building interiors from point cloud data. In: ICIP (2012)
6. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building Rome in a day. Communications of the ACM (2011)
7. Huang, Q.X., Anguelov, D.: High quality pose estimation by aligning multiple scans to a latent map. In: ICRA (2010)
8. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments. In: ISER (2010)
9. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: ICCV (2009)
10. Hernández Esteban, C., Vogiatzis, G., Cipolla, R.: Probabilistic visibility for multi-view stereo. In: CVPR (2007)
11. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. In: SIGGRAPH Asia (2009)
12. Suveg, I., Vosselman, G.: Reconstruction of 3d building models from aerial images and maps. ISPRS Journal of Photogrammetry and Remote Sensing (2004)
13. Uyttendaele, M., Criminisi, A., Kang, S.B., Winder, S., Szeliski, R., Hartley, R.: Image-based interactive exploration of real-world environments. CGA (2004)
14. CGAL: Computational Geometry Algorithms Library (2012), `http://www.cgal.org`
15. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3D architectural modeling from unordered photo collections. In: SIGGRAPH Asia (2008)
16. Delong, A., Boykov, Y.: A scalable graph-cut algorithm for n-d grids. In: CVPR (2008)
17. Garland, M.: Qslim: Quadric-based simplification algorithm (1998)