Noname manuscript No.
(will be inserted by the editor)

Reconstructing the World's Museums

Jianxiong Xiao · **Yasutaka Furukawa**

Received: date / Accepted: date

Abstract Virtual exploration tools for large indoor environments (e.g. museums) have so far been limited to either blueprint-style 2D maps that lack photo-realistic views of scenes, or ground-level image-to-image transitions, which are immersive but ill-suited for navigation. On the other hand, photo-realistic aerial maps would be a useful navigational guide for large indoor environments, but it is impossible to directly acquire photographs covering a large indoor environment from aerial viewpoints. This paper presents a 3D reconstruction and visualization system for automatically producing 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 with a Constructive Solid Geometry (CSG) representation consisting of volumetric primitives, which imposes powerful regularization constraints. We also propose several novel 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.

Keywords Photorealistic Indoor Maps · Museum · Large-scale Image-based Modeling · 3D Reconstruction

Jianxiong Xiao
Princeton University
E-mail: xj@princeton.edu

Yasutaka Furukawa
Washington University in St. Louis
E-mail: furukawa@wustl.edu

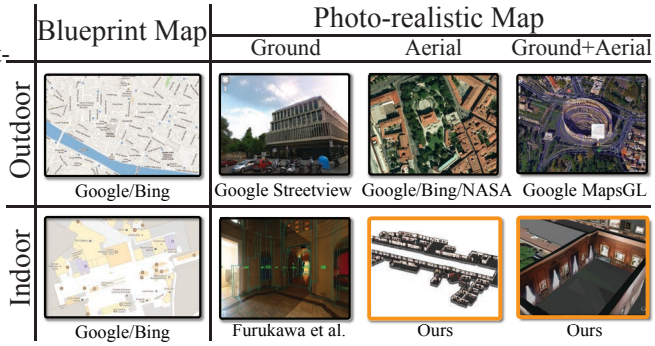

Fig. 1 Categorization of different map visualizations for indoor and outdoor environments. While major efforts have been made in the past for outdoor environments, there has not been much effort toward making photo-realistic visualizations of indoor scenes. This paper enables the photo-realistic visualization of indoor scenes from aerial viewpoints, which can be easily integrated with the conventional ground-based indoor navigation experiences.

1 Introduction

The abundance of photographic data combined with the growing interests in location-aware applications make 3D reconstruction and visualization of architectural scenes an increasingly important research problem with large-scale efforts underway at a global scale. For example, Google Maps 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 instance, the Google Art Project allows exploration of museums all over the world as well as close examination of hundreds of artworks photographed at high resolution. Google Maps and Bing Maps serve full-view panorama images of indoor businesses such as grocery stores and restaurants.

Fig. 2 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.

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 (See Fig. 1). 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. 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. Although we expect our method to generalize to other large indoor structures as well, we demonstrate our approach on large museums (See Fig. 2), since there is a clear need for a photorealistic navigation tool. Our system enables 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 (demos and videos available at [21]). 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. 3).

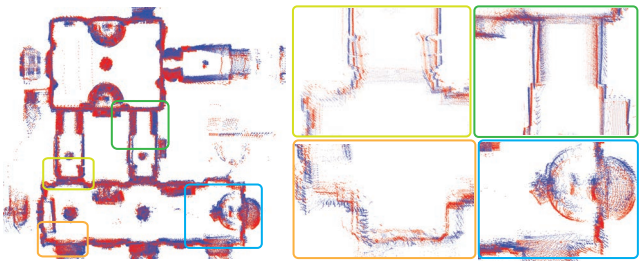

Fig. 3 A top-down view of input laser points, where two different colors represent two vertical laser range sensors. A challenge is the presence of severe noise in the input laser points. Note that errors are highly structured and "double walls" or even "triple walls" are prevalent.

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 [5,18], or the analysis of 3D structure such as the detection of symmetry or moldings [20,22]. In [19,24], a human-operated backpack system 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 [19], 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 [1,2], 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

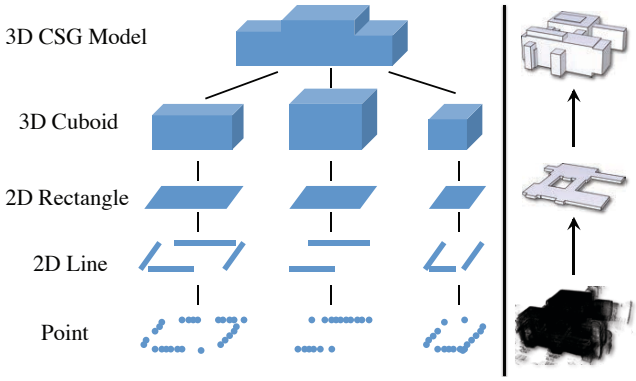

Fig. 4 3D CSG construction happens step by step in a bottom-up manner, from laser points, 2D lines, 2D rectangles, 3D cuboids to final 3D CSG models.

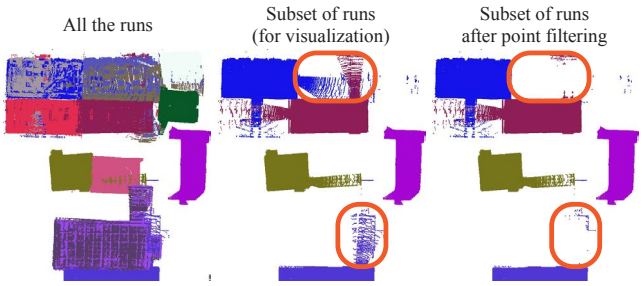

Fig. 5 Laser range sensors sometimes scan for a long distance, for example through doorways, which unfortunately yields sparse and noisy 3D points. The point filtering step removes such unwanted noisy points (highlighted in red).

systems. Active depth sensing usually yields more robust and accurate pose estimation [13,16]. However, even with high-quality laser scans, pose estimation is still very challenging [31], 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 grid [9, 14]. 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 [26,28,29] 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 [3]. 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. Single-view 3D reconstruction techniques [11,32,12,30,17] are also popular for architectural scenes, where priors and sophisticated regularization techniques play an important role, which is common in this paper. However, their focus is more on the analysis of 3D space, while our primary interest is scene visualization that requires higher quality 3D models.

For indoor scene visualization, view-dependent texture mapping is typically used to provide interactive scene exploration [9,27]. For example, Google Art Project, Google Maps, and Bing Maps provide immersive panorama-based viewing experiences for museums and indoor business locations. 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.

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 [16], and 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.

Laser range sensors on our trolley can scan for long distances, for example, rooms next door through narrow door ways (See Fig. 5). This causes a problem, because the same room can be scanned from multiple runs, where laser points from nearby runs are dense and accurate, while those from far runs become sparse and possibly inaccurate. Therefore,

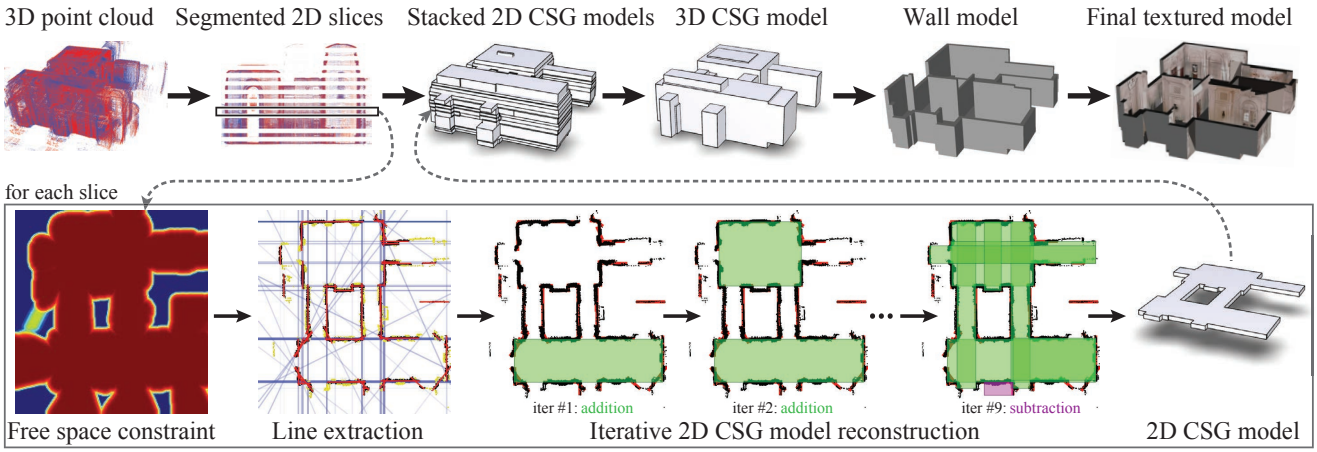

Fig. 6 System pipeline. The first row shows the entire pipeline of our system, and the second row shows the detailed pipeline of "Inverse CSG" reconstruction of a 2D horizontal slice.

we only keep laser points from nearby runs at overlapping regions. More concretely, given laser points from all the runs, we project them onto a horizontal plane, compute their bounding box, and overlay a grid of cells inside the bounding box, where the size of a cell is set to be 0.1 meters. At each cell, we identify $\alpha(=2)$ runs that have the most number of points in the cell, and filter out all the points that belong to the other runs. Note that this filtering procedure is not perfect and may leave unwanted points. However, our reconstruction algorithm is robust, and the main purpose of this step is to remove the majority of the unwanted point clouds.

3 Inverse CSG for Large-scale Indoor 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 [8,9]. 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) extract line segments from the input point clouds in a horizontal slide, which are used to form rectangle primitive candidates; (3) solve a 2D CSG model with rectangle primitives in each slice; (4) generate 3D primitives based on the 2D reconstructions, then solve for a 3D CSG model (See Fig. 4). We now detail in these steps.

3.1 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 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. 6, 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 [15]. These line segments are used to enumerate rectangle candidates in several ways (Fig. 8).

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

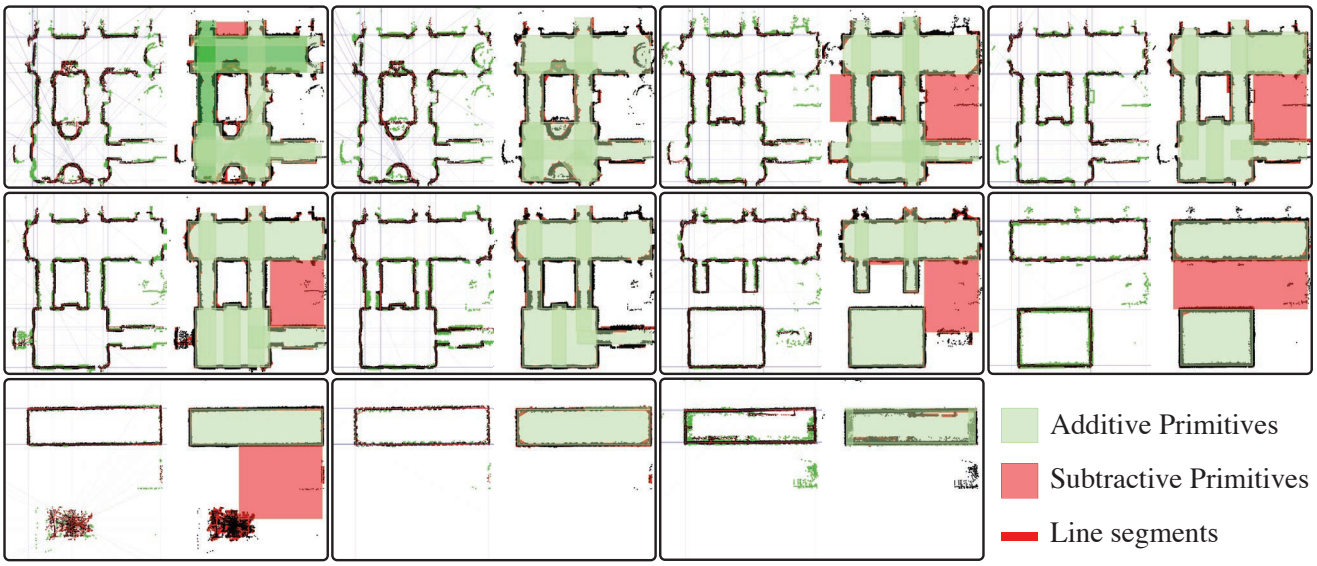

Fig. 7 2D CSG reconstructions for the eleven horizontal slices for one run in Metropolitan Museum of Art in New York City (Met). For each slice, the figure shows the 3D point cloud and extracted line segments (in red) at the left, and reconstructed 2D CSG model at the right, where green (resp. red) rectangles are additive (resp. subtractive) primitives.

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 (See bottom of Fig. 8).

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.

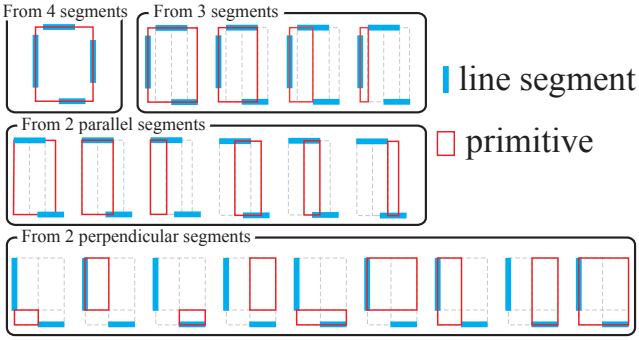

Fig. 8 Line segments are used to form rectangle primitives. Depending on the number of available line segments, different number of rectangle primitives are generated.

3.2 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 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

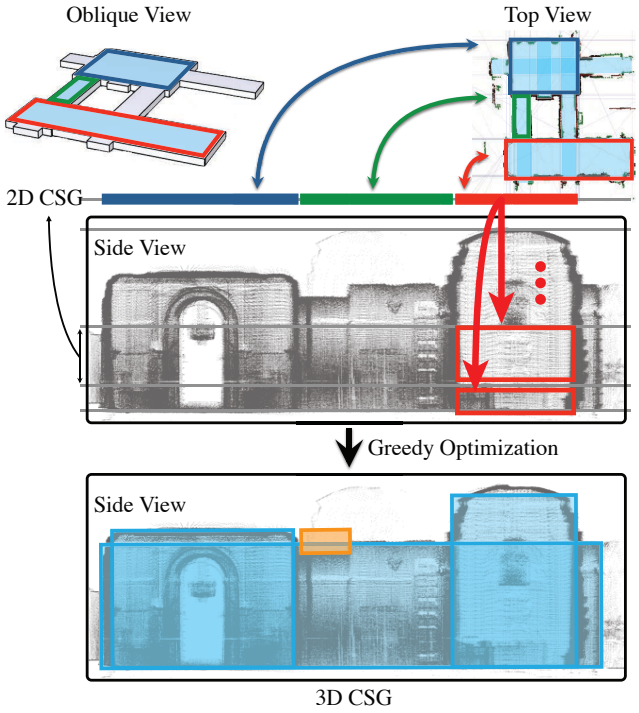

Fig. 9 From 2D CSG to 3D CSG. The first row contains the oblique and the top views of the 2D CSG model reconstructed in a horizontal slice, which consists of 2D rectangles. In the second row, each 2D rectangle is inflated to generate 3D cuboid primitives, where the top and the bottom faces are determined by using all possible pairs of horizontal slice boundaries. The figure shows two such cuboid primitive examples, which are generated from the red rectangle. In the third row, a pool of cuboid primitives are used to construct 3D CSG model, where we allow subtraction operations as in the 2D case, and a subtractive primitive is illustrated in orange.

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 number of 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 [7] 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

[1] Different from standard volumetric reconstruction algorithms [9, 14], voxel resolution is not critical for accuracy in our approach, as precise surface positions are determined by primitives.

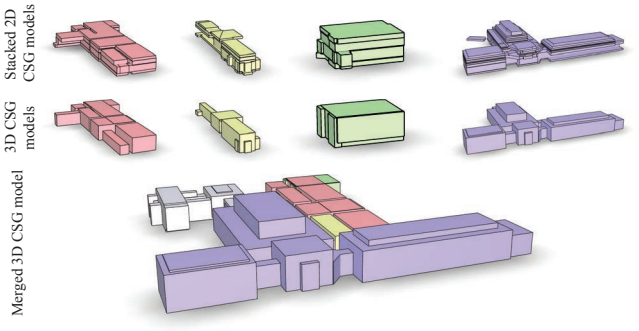

Fig. 10 Reconstruction results for the Frick Collection. Top: Stacked 2D CSG models for individual runs. Middle: Corresponding 3D CSG models. Bottom: The final merged 3D CSG model. The white model is the example illustrated in Fig. 6. Note that there exist three more runs for this museum, which are not visible here because of overlapping or occlusion in rendering.

2D slice, we simply take the average score value inside the slice vertically.

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:

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

Note that these three energy terms are normalized by their definition. The overall objective function $E(T)$ is defined to be a weighted sum of the three terms:

$$E(T) = w_1 E_1(T) + w_2 E_2(T) + w_3 E_3(T), \quad (1)$$

where $w_1 = 1$, $w_2 = 0.1$ and $w_3 = 0.4$. To decide the weights, we tried various combinations on several datasets. We found that our algorithm is not very sensitive to these parameters, and a reasonable combination will produce similar results.

3.3 Reconstructing 3D CSG Model

Having reconstructed a 2D CSG model for each slice, one possible way to generate a 3D CSG model is to extrude

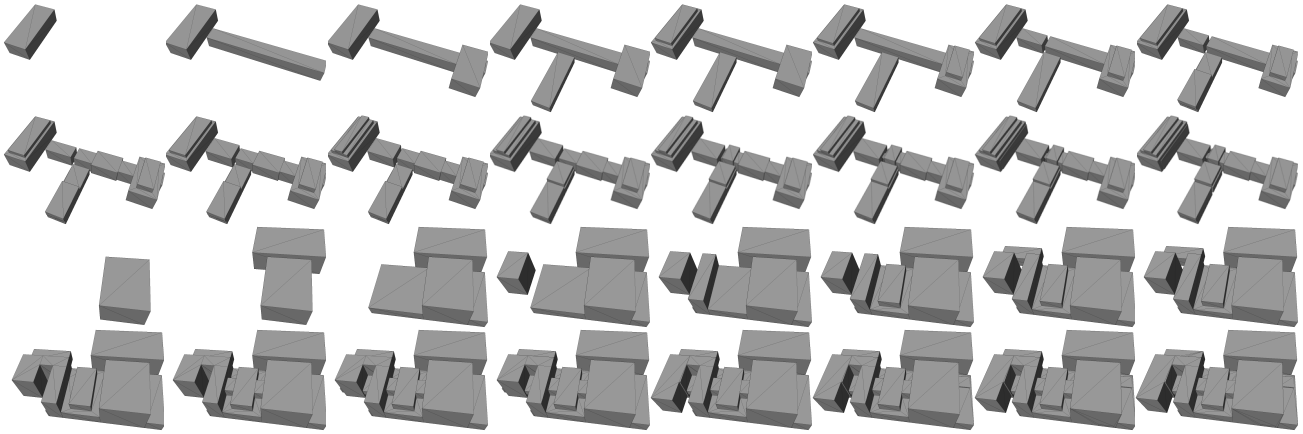

Fig. 11 Two examples of the 3D CSG optimization process. Our 3D CSG construction algorithm either adds or subtracts a cuboid primitive at each step, while greedily optimizing the objective function. Example model progression is shown for a run in National Gallery and Met, respectively.

each 2D CSG model as thick as the corresponding horizontal slice, and stack them up. However, this solution tends to create jagged, misaligned models (e.g. the first row of Fig. 10), as 2D CSG models are reconstructed independently in all the slices, which may be roughly consistent but are not exactly. To address this issue, we propose a 3D Inverse CSG algorithm to generate a 3D model, as shown in Fig. 9. We follow the same two-step InverseCSG algorithm for the 2D slice reconstruction, and extend it to handle 3D reconstruction: first to generate a pool of primitives (3D cuboids), and then use our greedy algorithm to choose a subset of primitives from the pool to represent the final complete shape.

For the first step, 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). Therefore, let N denote the number of the horizontal slice boundaries. Each 2D rectangle generates $\binom{N}{2}$ cuboid primitives. Then, we define the same objective function as before in Eq. 1, which can be naturally generalized to 3D, and use the same greedy algorithm as in Sec. 3.2 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 [4].

Intermediate reconstruction results are shown in Fig. 7 and Fig. 11. Fig. 7 shows 2D CSG reconstruction results for one run of Metropolitan Museum of Art in New York City (Met), which consists of eleven horizontal slices. Note that 2D CSG models are fairly consistent across different slices but are not exactly, and hence, a 3D CSG construction step is necessary to produce a clean and compact 3D model. Fig. 11 shows how the 3D CSG construction algorithm iteratively improves the model for runs in National Gallery and Met. Notice that a very small number of simple cuboids can represent fairly complex building structure.

[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.

4 Indoor Scene Visualization

Our goal in visualization is to display an entire museum from aerial viewpoints for effective navigation. However, a reconstructed CSG model 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. In this section, we first explain CSG manipulation techniques to remove ceilings and add thickness to walls for the construction of view-independent wall models. Second, we propose a technique to optimize the visibility of wall models for a given viewing direction, in particular, lower the back-facing walls to construct view-dependent wall models optimized for the view. Third, we introduce our scalable texture mapping algorithm, which can handle a very large mesh with tens of thousands of input images. Lastly, we explain our interactive indoor scene navigation systems that make use of the wall models.

4.1 View-independent Wall Model Construction

Denote a reconstructed 3D 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. 13).

In detail, let Δ be a manually specified thickness of the wall ($\Delta = 0.5$m). We inflate (resp. shrink) each additive (resp.

Fig. 12 View-dependent model construction can lower back-facing walls for increased visibility. Different height thresholds are used to generate models at the right. This example shows the close-ups of the State Tretyakov Gallery.

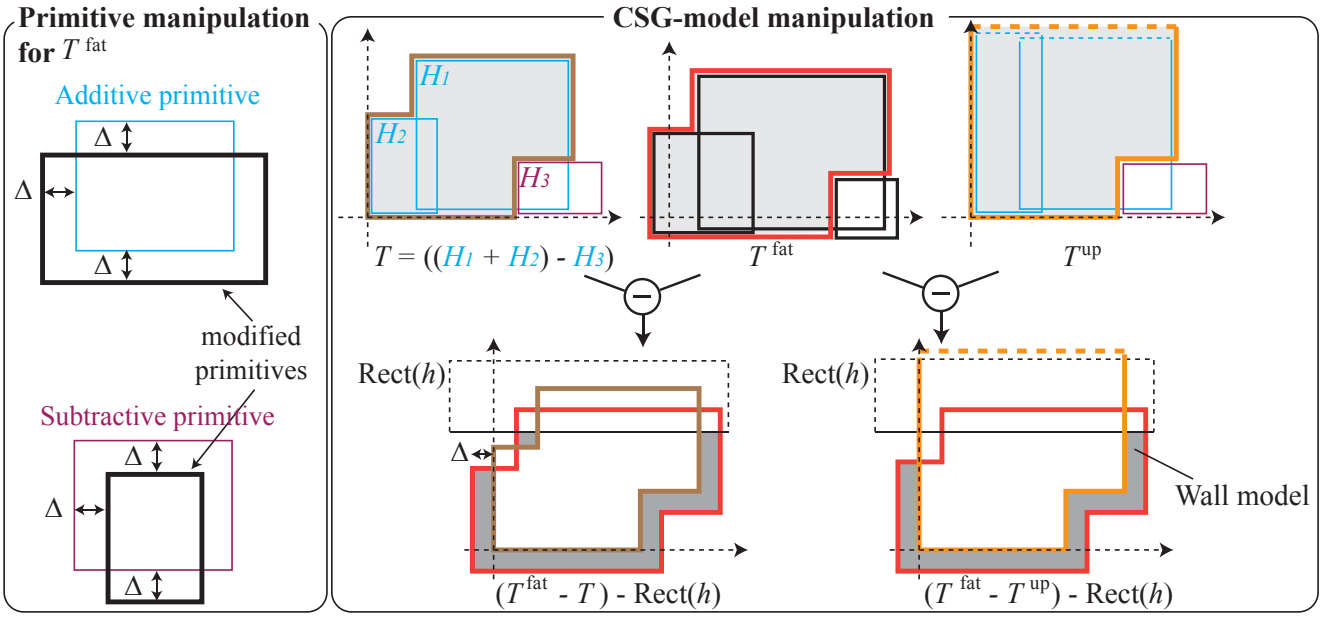

Fig. 13 Given a CSG model T, we expand each additive primitive and shrink each subtractive primitive (left), to generate an inflated model T^{fat}. Subtracting T from this inflated model T^{fat} gives us a wall model with thickness (right). To remove floating walls due to a complex ceiling structure, we subtract T^{up} that is constructed to have infinite height, instead of T.

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 Rect(h): $((T^{\text{fat}} - T) - \text{Rect}(h))$, where 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. 13, 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.

4.2 View-dependent Wall Model Construction

When a viewing direction is fixed in an application, we can further optimize the visibility of the model for the given direction by lowering back-facing walls. Fig. 14 illustrates how CSG model can be further manipulated to lower back-facing wall. Let v be the viewing direction, for which the model is to be optimized, and denote h_f and h_b as the maximum height of the structure to be reconstructed for the front-facing and back-facing walls, respectively. For each primitive cuboid, consider a local coordinate frame whose XYZ axes are aligned with the cuboid. Then, we translate the primitive along each of the XYZ axes along v direction by $\Delta(= 0.5\text{m})$. Let us denote this translated model as T^v, as shown in the second column of Fig. 14, then $(T^v - T^{\text{up}}) - \text{Rect}(h_f)$ generates front-facing facades with the height limited at h_f as in the third column of Fig. 14. Similarly, let $\overline{v}$ be the inverted reflection vector of v against the ground plane, then the back-facing facades are modeled by $(T^{\overline{v}} - T^{\text{up}}) -$

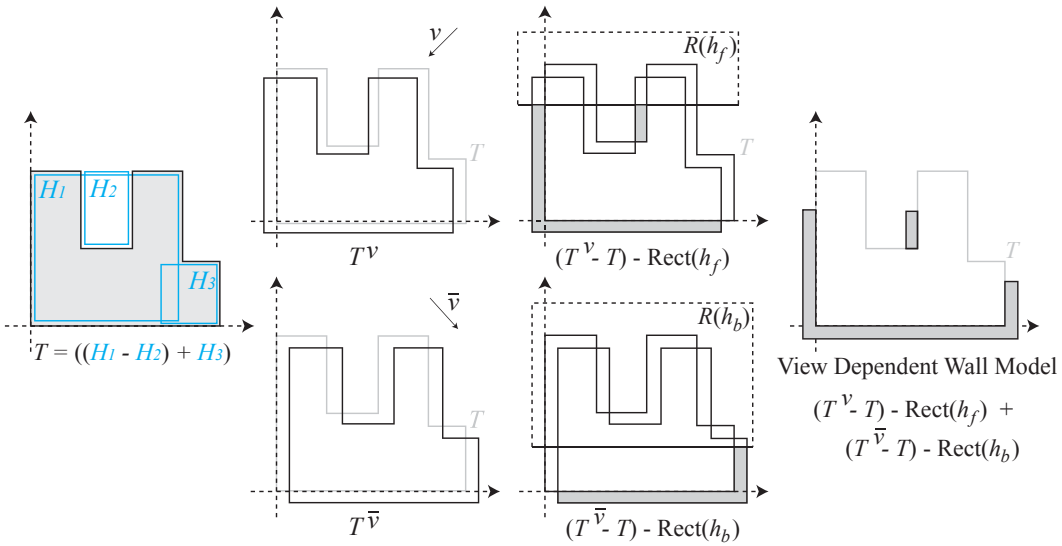

Fig. 14 View dependent model optimization. For an application with a fixed viewpoint, we can further manipulate 3D CSG structure to lower back facing walls. We construct front facing walls (top row in the middle columns) and back facing walls (bottom row in the middle columns) independently, which are merged by simply taking the union operation in the CSG representation to construct a view dependent wall model.

Rect(h_b). The finally view-dependent 3D model is obtained by taking their union:

$$((T^{v} - T^{\mathrm{up}}) - \mathrm{Rect}(h_f)) + ((T^{\bar{v}} - T^{\mathrm{up}}) - \mathrm{Rect}(h_b)).$$

h_f and h_b are set to 6 and 2 meters, respectively, in our experiments. Effects of the view-dependent construction on real examples are shown in Fig. 12.

4.3 Texture Mapping

The last step of our pipeline is to texture-map a wall model from registered photographs. Modern techniques minimize texture stitching artifacts over multiple faces simultaneously [25,29]. We take a similar approach but with modifications, as our system needs to be scalable to an entire building with tens of thousands of input images. It needs to be also robust against reconstruction errors in a 3D model as well as 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]. Note that stitching artifacts are expected to be present at face group boundaries, but this is not an issue in practice, because face group boundaries typically correspond to room and floor boundaries, where textures are not very important. While our CSG representation inherently contains such face group information, the library we used in our implementation – CGAL [4] – 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 [25] 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 $20 \times 20\mathrm{cm}^2$), then up-sample the resulting stitching information to generate higher resolution texture images (one pixel per $1\mathrm{cm}^2$). 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 [6], but it is still expensive. Our scheme exploits our compact 3D model, and allows easy implementation that works well in practice.

4.4 Ground View and Aerial View Navigation

A global texture-mapped mesh model enables visualization of large indoor scenes from aerial viewpoints, which is more effective for navigation and exploration than traditional ground-only (e.g., panorama-based) visualization. Our model can be rendered from arbitrary viewpoints, but is not effective for

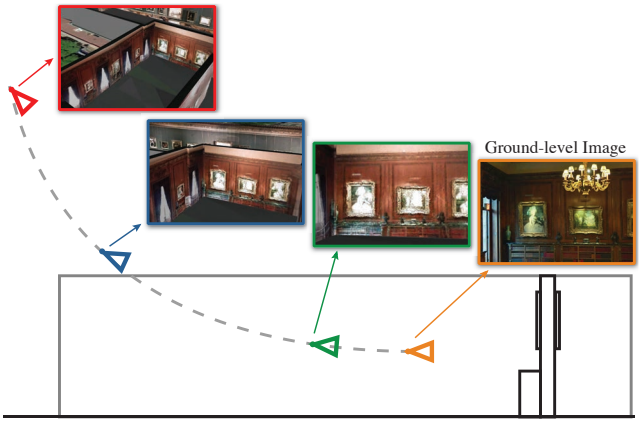

Fig. 15 Transition from bird's eye view to ground-level view. This enables an intuitive transition between two modes of visualization, and gives the map users a better sense of their current location.

close-range ground viewpoints, because the model only captures dominant facade geometry and lacks in small objects or any non-rigid structure such as water from fountain. For a ground viewpoint, a nearby input image is much more immersive, free from any artifacts, and can visualize even non-rigid objects and view-dependent effects such as non-diffuse reflections. However, ground level visualization alone is ineffective for navigation due to the limited visibility and mobility.

To integrate the merits from both modes and overcome their disadvantages, we propose to employ traditional panorama based navigation for ground viewpoints, while allowing users to switch between ground and aerial viewpoints at any time. In our implementation, we geo-register our 3D mesh models and load them to the Google Earth to control the camera path as in Fig. 15. Our 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. See our project website for the videos [21].

Another popular digital mapping implementation is tile-based visualization (See Fig. 16), where pre-rendered tile images are displayed to users in multiple resolutions, which allow intuitive panning and zooming operations. Standard top-down views are not effective for indoor scenes, where vertical facades contain important information but are not visible. Therefore, we pre-render our texture mapped model from four oblique viewing directions, corresponding to north, east, west, and south-headings, then load the rendered images as a set of tiles to Google Maps through Google Maps API. Note that this is a perfect example to make use of view-dependent wall models, where for each heading, a view-dependent model, optimized for the corresponding viewing direction, is used to render tiles. See our project website to try this demo [21].

Fig. 16 A global texture-mapped mesh model allows indoor scene visualization based on pre-rendered tiles, which has been a popular technique for online digital mapping products. We generate tiles for four different headings (i.e., north, south, east and west) in multiple resolution levels by using the view-dependent wall models, where back-facing walls are lowered to improve visibility for each direction. This figure shows the Uffizi Gallery from the two different headings.

5 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 2,670 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. 10 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. 6).

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

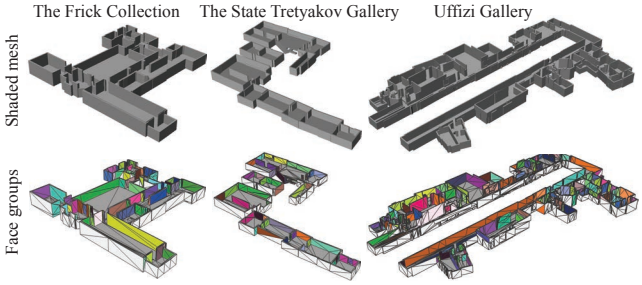

Fig. 17 Shaded renderings of the view-dependent models and the corresponding face-groups. A random color is assigned to each group except for the white faces that are not visible from any input camera and do not require texture stitching, and gray faces that are identified as floors.

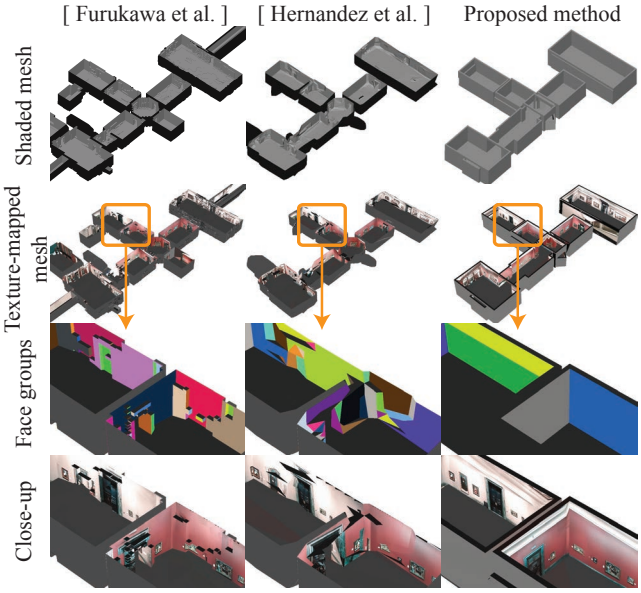

Fig. 18 Comparison with two state-of-the-art 3D reconstruction algorithms [9,14].

Figure 17 shows shaded renderings of the view-independent wall models and the extracted face groups, which illustrates that the algorithm successfully identifies major floor and wall structures as face groups. Figure 19 shows final texture-mapped models with some close-ups. The models in these two figures are obtained by the view-dependent CSG construction for better visibility. Figure 20 also shows final texture-mapped mesh model of Met with close-ups, which is the largest reconstruction in our experiments. The model is constructed by the view-independent method. These examples illustrate that the stitched textures are of high quality with very few artifacts. Note that our model focuses on dominant facades and does not capture small-scale details or objects. Nonetheless, we can often see and even recognize such missing structure and objects in texture-mapped imagery. Thanks to our regularized mesh model, an image usually goes through a simple affine transformation (assuming weak perspective) during texture mapping without much stitching. Our eyes are surprisingly good at correcting such low frequency distortions and understanding image contents. On the other hand, texture stitching, which is more necessary to fill-in texture for complicated meshes, causes annoying artifacts to our eyes. Our approach pushes stitching artifacts to facade boundaries and succeeds in producing high quality textures in the middle of each facade even where geometry is inaccurate. In an extreme case, at the bottom right in Fig. 20, our model misses an entire room, but the room can be recognized through texture-mapped imagery on a planar wall without major artifacts.

The effects of the view-dependent construction are illustrated in Fig. 12, 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. 18, we compare our al-

Fig. 19 Final view-dependent texture-mapped 3D models with some close-ups. More results are available in the project website [21].

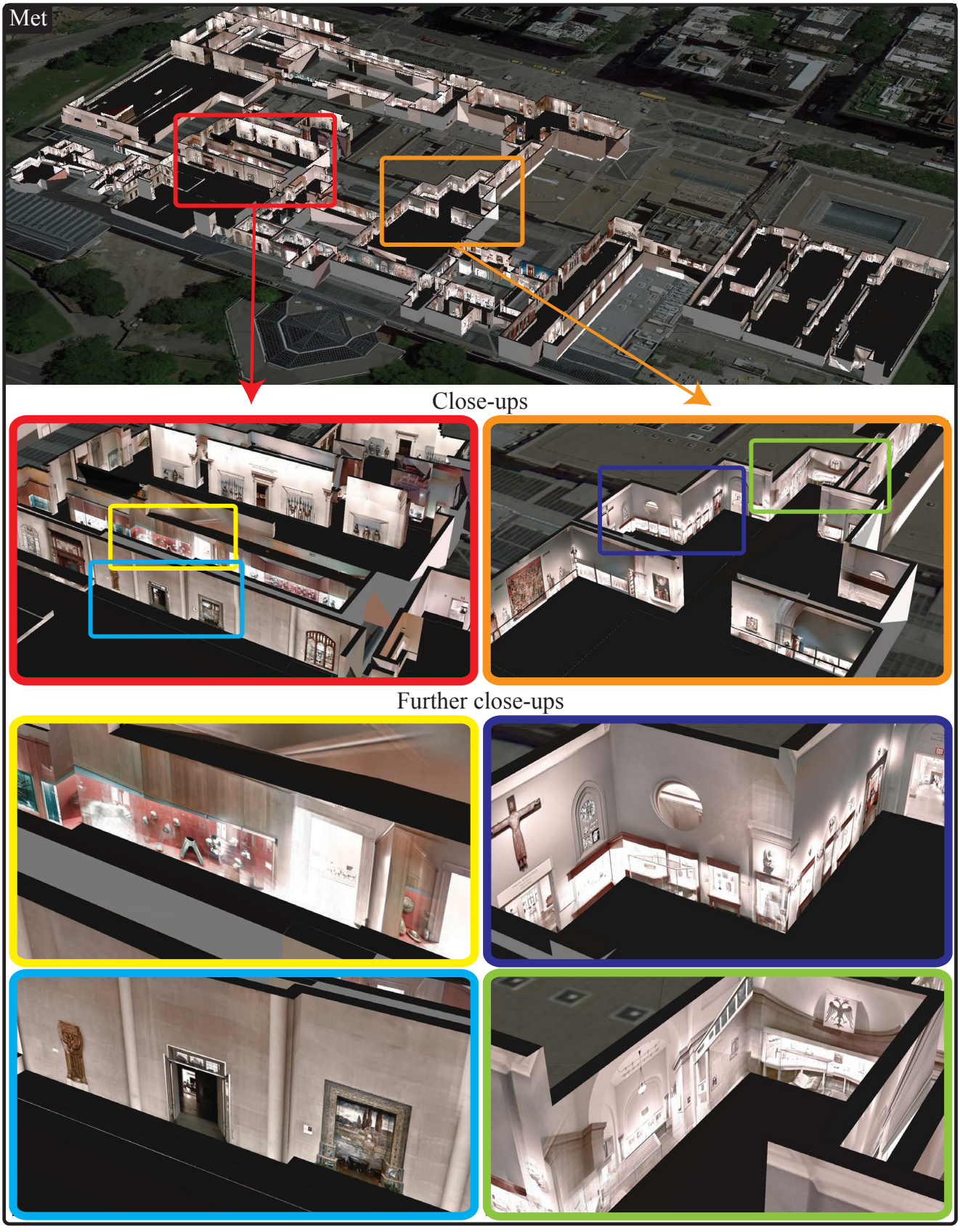

Fig. 20 The texture-mapped 3D model for The Metropolitan Museum of Art. In the close-up at the bottom right, the texture of an entire room is mapped to a plane due to the lack of precise geometry. The stitched texture is free from major artifacts, except for an inconsistent perspective.

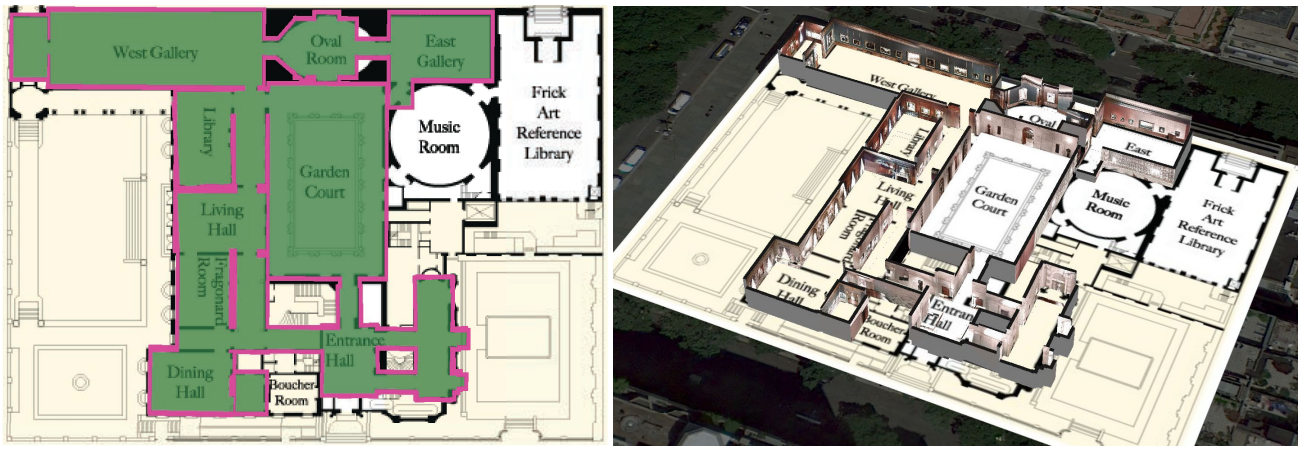

Fig. 21 Left: Manually overlaying our model with a floor plan image of The Frick Collection. For our model, floor is colored in green and walls are in pink. Right: This overlaying enables hybrid visualization where the texture-mapped model is rendered over the floor plan image and a (darkened) satellite imagery.

gorithm with two state-of-the-art techniques – Hernández et al. [14] and Furukawa et al. [9], both of which can usually tolerate noise better than a technique used by the Computer Graphics community [5]. 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 [14] 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 [10] 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. 18, 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. This is simply because the data capturing device did not enter those rooms, and 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 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. 21, we manually 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. 21: The texture-mapped model is rendered on top of the floor plan image, without face-groups identified as floors.

Lastly, 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. 19), 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. 21), or the hexagonal shaped room in National Gallery (Fig. 19). Texture mapping suffers from severe artifacts at such places as shown in the left of Figure 22. Input pose errors in cameras and laser points also yield severe artifacts as in the right of Fig. 22, where the geometry reconstruction completely fails.

6 Conclusion

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, and enables an intuitive transition between bird's eye view and ground-level view, which we believe opens up new possibilities for indoor scene navigation and mapping. More results and demo can be found in our project website [21].

Our future work includes extension of geometric primitive types to more shapes, such as cylinders or spheres. Our major failure modes are due to severe errors in the input pose information, and we need to explorer better pose es-

The Frick Collection The State Tretyakov Gallery

Fig. 22 Limitations of current approach. Left: Our structure assumption is more general than Manhattan-world, but cannot handle curved surfaces such as an oval room. Right: When input pose errors are significant, our geometry reconstruction completely fails.

timation algorithm. It is also interesting to model objects in a scene, such as sculptures, glass cabinets and tables that are currently missing in our models. Furthermore, although we restrict our demonstration on museums, it would be interesting to adapt our approach to other large indoor structures as well, given that high-quality RGB-D dataset for big spaces recently became available [31]. Another challenge would be to go beyond the notion of face-groups, and identify *objects of interests* (e.g. paintings), and associate semantic information to enrich the model [23].

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, Antonio Torralba, Bill Freeman and Andrew Owens for brainstorming and discussion. We thank anonymous reviewers for valuable feedbacks. This work was started when Jianxiong Xiao was an intern student at Google Inc. under the supervision of Yasutaka Furukawa.

References

1. 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)
2. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building rome in a day. In: ICCV (2009)
3. Autodesk: Revit architecture (2012)
4. CGAL: Computational Geometry Algorithms Library (2012). Http://www.cgal.org
5. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH, pp. 303–312 (1996). DOI http://doi.acm.org/10.1145/237170.237269
6. Delong, A., Boykov, Y.: A scalable graph-cut algorithm for n-d grids. In: CVPR (2008)
7. Felzenszwalb, P., Huttenlocher, D.: Distance transforms of sampled functions. Tech. rep., Cornell (2004). Computing and Information Science TR2004-1963
8. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Manhattan-world stereo. In: CVPR (2009)
9. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: ICCV (2009). DOI 10.1109/ICCV.2009.5459145
10. Garland, M.: Qslim: Quadric-based simplification algorithm (1998)
11. Gupta, A., Efros, A.A., Hebert, M.: Blocks world revisited: Image understanding using qualitative geometry and mechanics. In: ECCV (2010)
12. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: ICCV (2009)
13. 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)
14. Hernández Esteban, C., Vogiatzis, G., Cipolla, R.: Probabilistic visibility for multi-view stereo. In: CVPR (2007)
15. Hough, P.V.: Machine analysis of bubble chamber pictures. In: Proceedings of International Conference on High Energy Accelerators and Instrumentation (1959)
16. Huang, Q.X., Anguelov, D.: High quality pose estimation by aligning multiple scans to a latent map. In: ICRA (2010)
17. Jiang, H., Xiao, J.: A linear approach to matching cuboids in RGBD images. In: CVPR (2013)
18. 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)
19. 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)
20. Pauly, M., Mitra, N.J., Wallner, J., Pottmann, H., Guibas, L.J.: Discovering structural regularity in 3d geometry. SIGGRAPH (2011)
21. Project website: http://vision.princeton.edu/projects/2012/museum/
22. Rodriguez, E.V., Oliver, A.A., Huber, D., Cerrada, C.: Detection, modeling, and classification of moldings for automated reverse engineering of buildings from 3d data. In: ISARC (2011)
23. Russell, B.C., Martin-Brualla, R., Butler, D.J., Seitz, S.M., Zettlemoyer, L.: 3D Wikipedia: Using online text to automatically label and navigate reconstructed geometry. In: SIGGRAPH Asia (2013)
24. Sanchez, V., Zakhor, A.: Planar 3d modeling of building interiors from point cloud data. In: ICIP (2012)
25. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3D architectural modeling from unordered photo collections. In: SIGGRAPH Asia (2008)
26. Suveg, I., Vosselman, G.: Reconstruction of 3d building models from aerial images and maps. ISPRS Journal of Photogrammetry and Remote Sensing (2004)
27. Uyttendaele, M., Criminisi, A., Kang, S.B., Winder, S., Szeliski, R., Hartley, R.: Image-based interactive exploration of real-world environments. CGA (2004)
28. Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., Quan, L.: Image-based façade modeling. In: SIGGRAPH Asia (2008)
29. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. In: SIGGRAPH Asia (2009)
30. Xiao, J., Hays, J., Russell, B.C., Patterson, G., Ehinger, K., Torralba, A., Oliva, A.: Basic level scene understanding: Categories, attributes and structures. Frontiers in Psychology **4**(506) (2013)
31. Xiao, J., Owens, A., Torralba, A.: SUN3D: A database of big spaces reconstructed using sfm and object labels. In: ICCV (2013)
32. Xiao, J., Russell, B., Torralba, A.: Localizing 3D cuboids in single-view images. In: NIPS (2012)