Watertight Planar Surface Meshing of Indoor Point-Clouds with Voxel Carving

Eric Turner and Avideh Zakhor
Department of Electrical Engineering and Computer Science
University of California Berkeley
Berkeley, CA 94720
elturner@eecs.berkeley.edu, avz@eecs.berkeley.edu

Abstract

3D modeling of building architecture from point-cloud scans is a rapidly advancing field. These models are used in augmented reality, navigation, and energy simulation applications. State-of-the-art scanning produces accurate point-clouds of building interiors containing hundreds of millions of points. Current surface reconstruction techniques either do not preserve sharp features common in a man-made structures, do not guarantee watertightness, or are not constructed in a scalable manner. This paper presents an approach that generates watertight triangulated surfaces from input point-clouds, preserving the sharp features common in buildings. The input point-cloud is converted into a voxelized representation, utilizing a memory-efficient data structure. The triangulation is produced by analyzing planar regions within the model. These regions are represented with an efficient number of elements, while still preserving triangle quality. This approach can be applied to data of arbitrary size to result in detailed models. We apply this technique to several data sets of building interiors and analyze the accuracy of the resulting surfaces with respect to the input point-clouds.

1. Introduction

Point-cloud scans of building interiors are useful in the fields of architecture, civil engineering, and construction. It is often desirable to use these point-clouds to construct meshed surfaces for texturing or geometric analysis. Meshed triangulations allow for the efficient representation of the scanned geometry and can be used for virtual walk-throughs of environments, augmented reality, indoor navigation applications, and energy simulation analysis. These applications rely on the accuracy of a model as well as its compact representation.

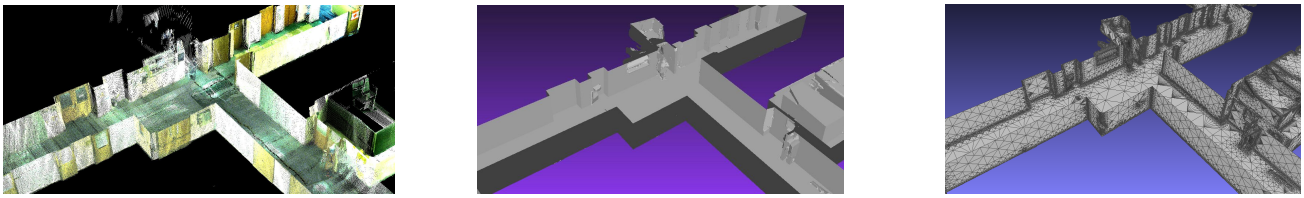
One of the primary challenges of indoor modeling is the sheer size of the input point-clouds. Scans of single floors of buildings result in point-clouds that contain at least hundreds of millions of points, often larger than the physical memory in a personal computer. Man-made geometry is typically composed of planar regions and sharp corners, but many conventional surface reconstruction schemes assume a certain degree of smoothness and result in rounded or blobby output if applied to these models [2, 3, 10, 11, 16, 18]. In addition to large flat regions, building interiors also contain many small details, such as furniture. A surface reconstruction scheme must be able to preserve these fine details while remaining robust to registration errors and noise from the input point-cloud. The point-cloud may also have gaps or missing data, but an output mesh must remain watertight. Lastly, all of these concerns should be addressed in an algorithm that exports models that use an efficient number of triangles.

We propose a scheme that meets all of these requirements. We partition space volumetrically into interior and exterior sets to ensure the boundary between these areas is watertight. This paper presents a method that applies this partitioning on a discretized voxel grid. The input point-cloud is used to separate the interior and exterior voxels based on a carving method. The resulting carved voxels are used to define a boundary representing the surface. This boundary is segmented into planar regions, which are in turn triangulated. The resulting watertight surface sharply represents large features such as walls, floors, and ceilings while still preserving fine detail such as furniture and staircases. The surface is adaptively triangulated with an efficient number of high-quality triangles.

Section 3.1 describes the voxel carving method used to define interior and exterior voxels. Section 3.2 describes the surface reconstruction approach that fits planar regions to the carved voxels, and triangulates them efficiently. Section 4.1 shows qualitative results of the output, while Section 4.2 demonstrates the accuracy of the computed mesh.

2. Background

The state-of-the-art surface reconstruction techniques applied to building architecture often do not employ a vol-



(a) (b) (c)

Figure 1. Example processing of office building interior. Ceiling has been removed for visualization: (a) Input point-cloud; (b) Output surface; (c) Triangulation of surface.

umetric approach. These methods commonly assume that building geometry is piece-wise planar, with the orientation of planar elements as either perfectly horizontal or vertical. This assumption allows for plane-fitting to be performed on the input point-cloud, either by a histogram approach or random consensus [1, 23, 26]. Such approaches do not guarantee watertightness of the resulting mesh and can require substantial post-processing. While similar techniques exist that ensure watertightness, they are unable to capture fine details [29]. Such methods often have difficulty preserving the correct genus or connectivity of the final mesh and fail to capture the locations of doorways or short hallways. The carving method described in this paper results in much higher detail.

There exist techniques that attempt to mitigate the above factors for architecture modeling, but they often require computationally expensive global optimizations [6]. Those approaches work well for a limited modeling environment, but do not scale well. The largest tested model in [6] consists of 3.3 million points, whereas the example models in this paper contain 13 million to 115 million points. The desired technique is one that uses a volumetric approach to ensure watertightness and preserves sharp, planar features as well as fine detail, but is fast and memory efficient even with large models. An alternate approach to generating models of high detail is to use a classification scheme on the input point-cloud. Such schemes are capable of preserving the fine detail in the model, such as staircases [23] or furniture [17, 21, 24]. Unfortunately, these techniques are heavily dependant on the variance of the database of shapes available. Any mislabeling causes errors in the output mesh.

There have been several algorithms that reconstruct surfaces from point-clouds using a volumetric approach [2, 18]. These methods compute a Delaunay Tetrahedralization of the input points and use those simplices to partition space into interior and exterior domains. Since the output surfaces of these schemes are composed of a subset of the original points, the size of the generated model scales with the density of the input point-cloud. Further, these methods assume that the point-clouds are modeling smooth and continuous surfaces, which is not the case in building modeling. These algorithms may also require a global optimization step [18]. While advancements have been made to perform these computations in an efficient and out-of-core manner [8, 13], the resulting models are too large to be practical for graphical or simulation applications.

Algorithms such as Poisson Surface Reconstruction allow the user to specify a resolution parameter for the generation of more compact models [16]. These schemes guarantee watertightness by using an implicit surface to model the point-cloud [12]. While these approaches can be applied to large models using distributed computing techniques [4, 5], they are unsuited for modeling man-made architecture. The output models of these methods lack sharp features because they generate implicit surfaces using Gaussian basis functions. Additionally, many common triangulation schemes for implicit surfaces result in uniform elements [15, 22], which are undesirable for large, flat surfaces that can be modeled just as accurately with fewer elements. If these approaches are used on a discretized voxel grid, undesirable artifacts of the discretization are preserved, requiring the final mesh to be smoothed, thus reducing accuracy [10]. Algorithms that adaptively mesh an isosurface or simplify an existing mesh rely on the local feature size of a model [9, 11, 20, 31]. Models with flat regions or sharp corners, where the curvature approaches zero or infinity, can become degenerate or have poor quality. The goal of this paper is to create models that are composed entirely of such areas, so these techniques are not appropriate.

Models of building interiors are rich with flat surfaces and right angles. This prior knowledge supports the use of primitives that have these same aspects. Examples include voxel and octree structures, which are used in many carving techniques [3, 10, 25, 28, 30]. The advantage to such approaches is that they are robust to noise and registration errors in the input point-cloud. One of the challenges with voxel representations is memory and computational intensity, thus becoming tractable only when performed in a distributed or parallel fashion [32]. Some voxel carving approaches can also inadvertently remove small details [10]. This paper modifies voxel carving to address these issues and introduces memory-efficient data structures that produce models that preserve fine details with an efficient number of elements.

3. Approach

In this paper we consider surface reconstruction of 3D models of building interiors that are dominated by piecewise planar geometry. These models can be acquired with a mobile scanning device that traverses the hallways and rooms of a building. The example input point-clouds used in this paper are produced from an ambulatory scanning system mounted on a human operator as a backpack [7]. This system has three Hokuyo UTM-30LX laser range finders that scan along the plane orthogonal to the direction of motion. This orientation allows for detailed scans of local geometry as the operator walks by [10, 27]. The laser scans and camera imagery are used to recover the location of the scanner within the building at each timestep, allowing for a full point-cloud of the geometry to be constructed [19]. An example point-cloud is shown in Figure 1a.

Our proposed method computes a meshed surface from this point-cloud using the following steps. First, The point-cloud is used to determine the locations in the volume that are *interior* and *exterior* via a voxel carving scheme. We introduce a novel data structure that allows the carving to be computed in a memory efficient and scalable manner. Second, once interior and exterior voxels are labeled, the surface defined between these two labelings is segmented into planar regions, as shown in Figure 1b. Each region is meshed with triangles that are proportional to its size, as shown in Figure 1c. The result accurately and efficiently depicts the geometry of the building.

3.1. Voxel carving

This interior/exterior volume classification is performed on a voxel grid. Given an input resolution size r, each voxel is a cube whose sides are length r. Initially, all voxels are assumed to be *exterior*, referring to any space outside of the scanned volume or space that is represented by solid objects. The process of *carving* refers to relabeling a voxel from exterior to interior, which occurs when a voxel is found to intersect the line segment from the scanner to a corresponding scan point. If a laser passes through a voxel, that voxel is considered interior space.

For each laser scanner, there exists a track in space that represents the scanner's movement during data collection. This track is represented by a sequence of positions $T = (\vec{t}_1, \vec{t}_2, \vec{t}_3, ..., \vec{t}_N)$, where N is the number of locations sampled during data collection. These track samples are shown as purple circles in Figure 2a.

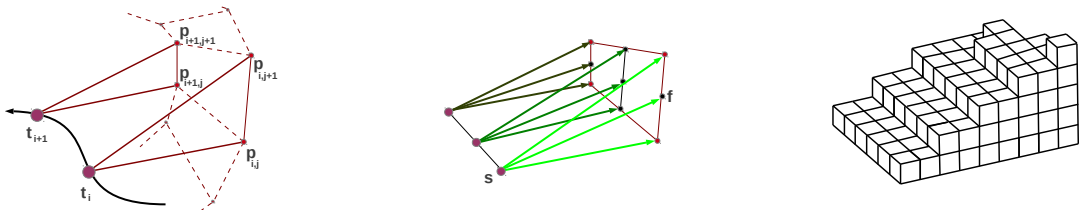
At the i^{th} timestep, each scanner sweeps an arc defined by the set of points $P_i = \{\vec{p}_{i,1}, \vec{p}_{i,2}, ..., \vec{p}_{i,j}, ..., \vec{p}_{i,M}\}$, where M is the number of samples along the arc. Each scanline is shown in solid red in Figure 2a. These scanlines can be interpolated in two dimensions, indexed by i and j. The first interpolates the laser scans temporally, while the second interpolates the scans spatially along the scan arc. These interpolations are shown as dashed lines in Figure 2a. By performing bilinear interpolation, a continuous surface of scans can be estimated from each scan point $\vec{p}_{i,j}$, shown as the interior of the quadrilateral ($\vec{p}_{i,j}$, $\vec{p}_{i,j+1}$, $\vec{p}_{i+1,j+1}$, $\vec{p}_{i+1,j}$). To efficiently determine which voxels are intersected by this interpolation, the carving operations are performed using ray-tracing between interpolated scanner position $\vec{s}$ and interpolated scan point position $\vec{f}$. Each pair $(\vec{s}, \vec{f})$ denotes a line segment to carve. An example set of these segments is shown in green in Figure 2b. By spacing these segments no more than distance r apart, each voxel within the interpolated volume is assured to be carved. We perform ray-tracing on each segment and relabel every intersected voxel as *interior*. This step produces a set of voxels as shown in Figure 2c.

Voxel data structure In most common voxel representations, each voxel in 3D space is explicitly stored in an array in memory. Even though this approach is straightforward and easy to use, its memory usage is proportional to the volume represented. For sizeable models, this memory footprint rapidly becomes intractable, necessitating splitting models into smaller chunks and processing each separately [10]. This step adds redundant computation and storage overhead. Adaptive approaches such as octrees reduce memory consumption by only representing the subset of relevant volume, but they still explicitly represent volume, an approach that rapidly fills memory [3, 30].

Rather than storing all relevant voxels in memory, in this paper we propose a data structure that implicitly represents the interior and exterior voxels by only explicitly storing the boundary voxels. A boundary voxel is defined to be one that is labeled as exterior, but has at least one face incident to a voxel labeled interior. The number of boundary voxels is proportional to the surface area of a model, so storing the boundary only requires $O(n^2)$ memory, whereas the full volume would require $O(n^3)$ memory to store, where n is the characteristic length of a model.

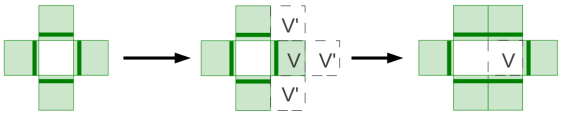
The data structure used during carving is a map between boundary voxel locations $v \in \mathbb{Z}^3$ and six boolean flags $(f_1, f_2, ..., f_6) \in \{\texttt{false}, \texttt{true}\}^6$, with the following invariants. Each of these flags represents one of the six faces of the referenced voxel. Marking $f_i = \texttt{true}$ indicates that the neighboring voxel of v that shares face f_i must be interior. If $f_i = \texttt{false}$, then this neighboring voxel is exterior, which may mean it is also a boundary voxel.

Figure 3 demonstrates in 2D how a voxel representation of the full model can be built from a starting configuration using ray-tracing as a primitive operation, while still respecting the above invariants. The starting configuration for the 2D map is shown in Figure 3a, with a single interior voxel represented using four boundary voxels. This interior voxel is initialized to be at the scanner's start posi-



(a) (b) (c)

Figure 2. (a) The input point-cloud is used in conjunction with the track of each scanner to define interior space to carve; (b) Carving is performed using ray-tracing from scanner location to an interpolation of the input points; (c) The result is a set of voxels labeled as *interior*.



(a) (b) (c)

Figure 3. A 2D example of carving a voxel. Stored boundary voxels are shown in green. White voxels are not explicitly stored. (a) The initial map configuration; (b) Voxel v is carved by removing v from the map and adding additional boundary voxels v' to the map; (c) v is represented as interior volume.

tion, which is known to be interior. Dark green lines indicate faces marked as `true`. Recall that interior voxels denoted in white are not explicitly stored in the map while the boundary voxels, denoted in light green, are stored explicitly. During the carving process, if a voxel v is designated to be carved then any of its faces that are flagged as `false` must be incident to exterior voxels, as shown in Figure 3b. Each of these neighboring exterior voxels, v', is added to the map, as they are now boundary voxels, and the face of v' that is incident to v is flagged as `true`. Lastly, v is removed from the map, which now represents that v is part of the interior volume, as seen in Figure 3c. Any carving attempt on a voxel that is not in the map can be ignored, since all carving initiates from within the interior volume. By using only this operation, the map invariants are preserved and will always consistently define an interior volume.

Preserving fine detail This carving process may not preserve features for point-clouds with low noise, but high detail. Specifically, objects whose feature length is on the order of one voxel size may be carved away. This issue can be a serious problem if two rooms are separated by a thin wall. Scanning both of these rooms may carve away this wall, resulting in a final model that shows only one, double-sized room. In order to preserve these features, we store a second voxel set that specifies the voxels that are intersected by the input point-cloud. While the original point-cloud is often much too large to be stored in memory at once, this discretization is much smaller and is on the same order of memory usage as the map of boundary voxels.

During the carving of each line segment, if ray-tracing encounters a voxel that is marked to contain input points, then the carving of that segment is truncated just before this voxel. No features that are represented in the point-cloud are ever carved away. Since ray-tracing already occurs on a voxelized grid, this occlusion check does not add any appreciable complexity to the computation.

Memory usage Data storage is an important factor in our algorithm. While the scheme described above only requires a small subset of the point-cloud to be in memory at any given time, it is also important to make sure that the intermediary and output data structures are reasonably sized. Figure 11 shows a moderate-size model depicting the corridors in a hotel, represented with a 6.3 GB point-cloud. The voxel carving, at a resolution of 5 cm, requires 12.7 MB of memory. If a conventional voxel grid structure were used to represent the entire bounding box, then 94.4 MB of memory would be required at this resolution.

3.2. Surface reconstruction

Our procedure for surface reconstruction of voxels can be broken into two parts. First, estimates of planar regions are found around the boundary faces of these voxels. These regions are formed from connected sets of voxel faces, all of which are positioned on best-fit planes. Second, each region is triangulated, forming a mesh. This triangulation lies along the best-fit plane for each region, with elements whose sizes are proportional to the size of the region.

Region growing on voxel faces The first task is to determine the connectivity along the carved voxel faces. An example of such a carving for a flight of stairs is shown in Figure 4a. Since these faces are squares that form a watertight surface and lie on an axis-aligned grid, each face has exactly four neighbors. If a voxel face and its neighbor are both oriented in the same direction, *e.g.* both have normal vectors in the Z+ direction, then one can immediately perform a flood-fill operation in order to group these faces into

planar regions. The faces belonging to each region lie exactly on a plane. The results of this flood-fill operation is shown in Figure 4b.

Since the voxels are a discretized representation of the volume, any flat surface of the environment that is not axis-aligned is represented as a zig-zag pattern of voxels. By fitting planes that only approximate the voxel faces, the output model can contain surfaces that are not axis-aligned. The approximating planes are found by performing Principle Component Analysis (PCA) on connected subsets of voxel faces [14]. For any connected set of voxel faces V, PCA is performed on the four corners of all the faces to estimate a best-fit plane. If V is well-modeled by this plane, then the elements of V are grouped together as one planar region. V is considered well-modeled if the maximum distance of V from the plane is at most r. This threshold guarantees that any voxels intersected by the modeling plane are incident to the faces in V.

Starting with the regions found in the flood-fill operation above, adjacent regions of voxel faces are progressively merged by attempting to model their union with a single best-fit plane. If this plane meets the threshold described above, then the two adjacent regions are replaced by one region representing their union. This step is referred to as *region growing*. Even though this stage reduces the total number of regions, it typically results in an over-fitting of too many regions. An example of this stage is shown in Figure 4c.

In order to yield a more aesthetically pleasing output, we further relax these region definitions. If two adjacent regions are fit by planes whose normal vectors are within $15°$, then they are replaced by a single region defined by their union. The result of this processing yields plane definitions that closely resemble an intuitive labeling of the floors, walls, and ceilings. This final region labeling is shown in Figure 4d.

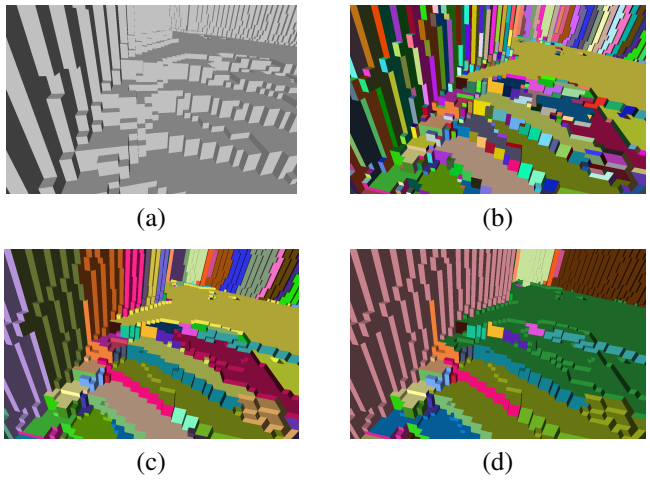


Figure 4. (a) Example carved voxels at the top of a flight of stairs; (b) Regions colored based on voxel face flood-fill; (c) Region growing by finding best-fit planes to voxel faces; (d) Regions relaxed to merge planes that are nearly parallel

Triangulation of regions Once the set of voxel faces has been partitioned into planar regions, it is necessary to triangulate these regions. Since the output mesh represents the planar regions found in the previous section, an optimum approach would adapt the size of triangles based on the size of these planar regions.

Taking advantage of the existing voxel grid helps ensure that each region is represented with good quality triangles. This grid allows for regions to be triangulated with a 2D variant of Isosurface Stuffing techniques, which provide strict bounds on resulting triangle angles [20]. An example region of voxel faces is shown in Figure 5a. Since this region is best-fit by a plane that is not axis aligned, the region is composed of voxel faces in a zig-zag pattern. The voxel faces that are most aligned with the normal vector of the region's plane, shown in red, are considered the dominant faces of the region. These dominant faces are projected along their corresponding axis to generate an axis-aligned 2D projection of the region. This projection is shown with black dashed lines in Figure 5a. The triangulation is found by populating a quadtree that is aligned to the projected grid with the faces of this region. An example of this quadtree is shown in Figure 5b. The tree is triangulated by placing vertices at the center and corners of the leaf nodes, as shown in Figure 5c. This step results in larger triangles for larger leaf nodes, while still controlling the quality of the output triangles. This triangulation is projected back onto the plane defined by the region, to result in triangulated representation of this region in 3D space.

Since the connectivity between voxel faces is well-defined, the connectivity of the output triangulation is also well-defined. To ensure that the borders between planar regions are represented sharply, the vertices that are shared by multiple regions are snapped onto the intersection of those regions. This fits the intersection between two regions to a line, and the intersection of three or more regions to a point in space. This step yields a watertight mesh across regions, as can be seen in the intersection of three regions at the corner of a room in Figure 5d. To limit self-intersections in the final surface, the vertices that are shared by multiple regions are allowed to be displaced up to a distance threshold from their original position in the voxel grid. This threshold is relaxed as the angle between the regions in question approaches $90°$. The corners between walls and ceilings remain sharp, while the transition between regions that are close to parallel is smooth. If such a threshold did not exist for the boundaries between near-parallel regions, their shared vertices could produce undesirable artifacts.

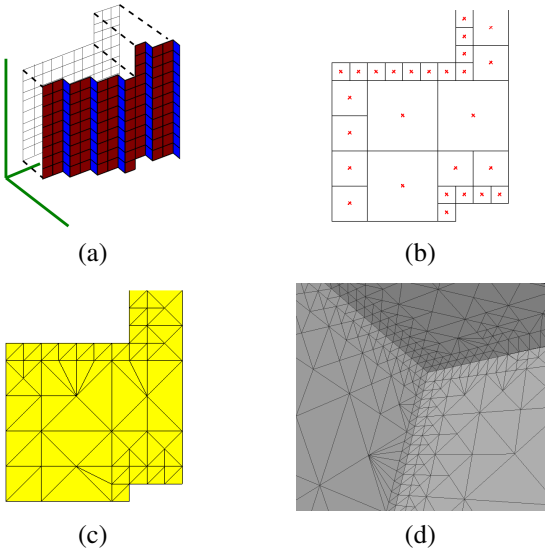


Figure 5. (a) The dominant faces of a planar region (shown in red) are projected to the dominant axis-aligned plane; (b) Projected faces represented in a quadtree structure to reduce number of elements; (c) This quadtree can be triangulated efficiently while ensuring high-quality triangles; (d) An example output of the triangulation of three regions in the corner of a room.

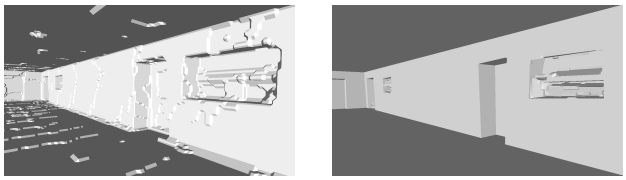


Figure 6. A visual comparison between (a) an existing voxel carving method [10] and (b) the proposed method at 5 cm resolution.

4. Results

The results of our surface reconstruction procedure are analyzed quantitatively and qualitatively. For quantitative analysis, the resulting mesh is compared to an existing voxel carving scheme, which uses Marching Cubes to generate a final output [10]. Figure 6 shows a qualitative comparison of the two schemes.

4.1. Example output meshes

The proposed algorithm was run on several datasets, which range in size from a single conference room to full floors of buildings such as hotels and shopping malls. The results are shown for sections of these models, along with the corresponding views of the original point-clouds. For these models, large flat areas are represented by fewer, larger triangles.

Figure 7 shows the reconstruction of a shopping mall's food court. The input point-cloud contains significant noise due to the amount of glass surfaces in the model, since most storefronts in the mall are glass. In the food court, the restaurants are well-modeled, as well as the ceiling and skylights. Figure 8 shows our largest model, with 220 million points. This model represents the aisles of a retail store, covering an area of $112.2\text{m} \times 77.5\text{m}$ using 2.7 million triangles. A smaller dataset is shown in Figure 9, representing a $10.5\text{m} \times 9.5\text{m}$ conference room with a hexagonal table in the center. This table, along with the podium to the left, is well-represented in the output. The ceiling of the conference room is inset with hanging lights, which can also be seen in the model. Figure 10 shows a modeling of a construction site. The surfaces of the room are represented by large regions, while the detail of objects is preserved. Lastly, Figure 11 shows an example of the full extent of a

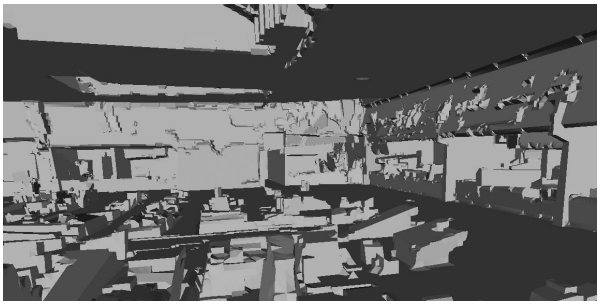


Figure 7. Surface reconstruction of a shopping mall. Resolution is 10 cm.

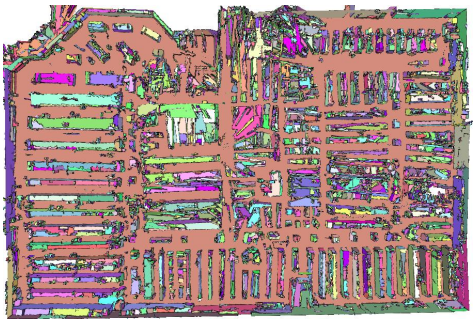


Figure 8. Surface reconstruction of a warehouse-sized retail shopping center, shown from top-down. Each planar region is given a random color. Resolution is 10 cm.

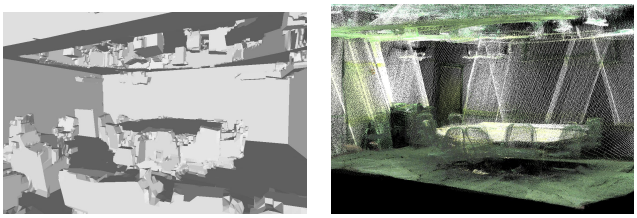
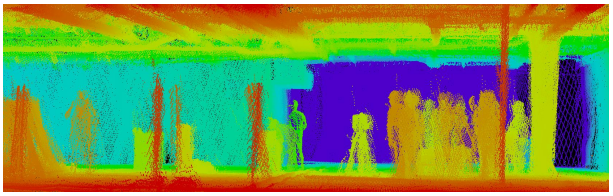
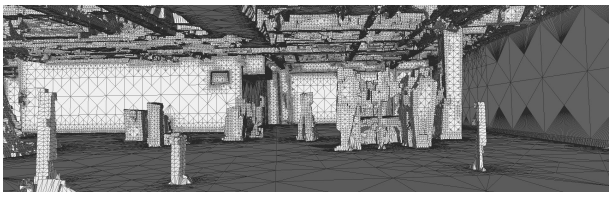


Figure 9. (a) Surface reconstruction of a $10.5\text{m} \times 9.5\text{m}$ conference room with table; (b) corresponding input point-cloud. Resolution is 5 cm.

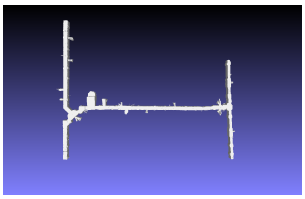


(a)

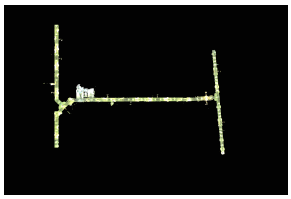


(b)

Figure 10. (a) Point-cloud of a construction site, colored by depth; (b) Triangle elements of generated surface. Resolution is 5 cm.

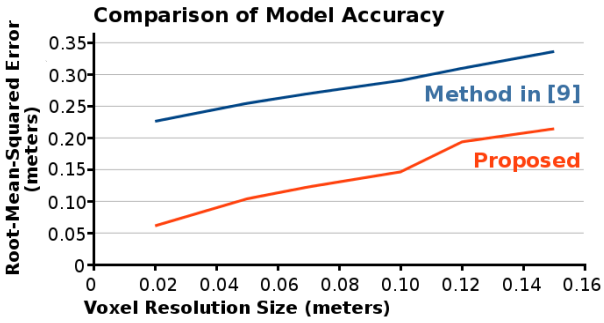


(a)

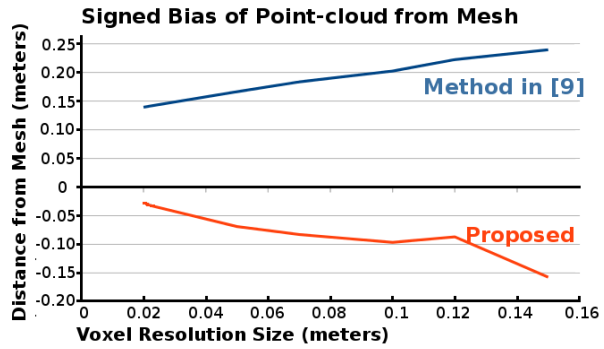


(b)

Figure 11. (a) Surface reconstruction of hotel hallway, showing full top-down view; (b) Corresponding full top-down view of input point-cloud.



(a)



(b)

Figure 12. (a) The root-mean-squared error and (b) bias of the input point-cloud with respect to the output of surface reconstruction schemes. This plot compares the proposed method of this paper with the voxel carving approach in a previous method [10].

$96.7\text{m} \times 75.7\text{m}$ H-shaped hotel hallway. The input point-cloud has 84 million points while the output model contains 933,000 triangles grouped into 3,096 planar regions.

Run-time analysis was performed on the dataset shown in Figure 10. The input to this dataset contains 25 million points. The code was run on a personal laptop with an Intel i7-2620M processor. The voxel carving, at 5 centimeter resolution, took 55 minutes of processing time. The surface reconstruction of these voxels took 1 minute and 2 seconds. Previous voxel carving schemes processed similar models of 15 million points in 16 hours at the same resolution [10]. Computation time was recorded for this same dataset with a resolution of 2 cm. Voxel carving took 12 hours and 10 minutes for this resolution, while surface reconstruction took 9.5 minutes.

4.2. Mesh error analysis

Accuracy of the output mesh is evaluated with respect to the input point-cloud. The distance of each input point to the nearest position on the output mesh is computed. For a given model, the root-mean-squared error is computed across all input points. Many fine details of the point-cloud cannot be represented perfectly in the final mesh, since they are the size of one voxel or smaller. As a result, even a perfect surface reconstruction of the voxels has finite error with respect to the point-cloud. By fitting regions directly to the voxels rather than triangulating with Marching Cubes, the error of the output surface can be mitigated. As shown in Figure 12a, at all resolutions the RMS error of the proposed method is lower than that of previous carving scheme [10].

A positive bias indicates an input point is inside the carved volume, while a negative value indicates a point is outside. As shown in Figure 12b, the proposed method yields a negative bias, since all carving is stopped before the input points are reached, ensuring no detail is carved away. The method in [10] carves through to the voxel containing the points, so its bias is positive and many small features are removed due to over-carving. Note that the absolute value of the bias at each resolution is lower for our proposed method than the method in [10]. This analysis was performed using the model shown in Figure 9.

5. Conclusion

This paper provides a mechanism for converting a point-cloud into a watertight triangulated mesh, with special consideration to modeling planar regions. Using a voxelized volumetric representation allows for a watertight output. By performing planar fitting on the input voxel faces be-

fore triangulation, the triangles can be adapted to the best-fit planes, resulting in fewer elements. This algorithm is novel in that it preserves sharp features, and the memory usage and computational performance of the method presented is favorable compared to other approaches.

Although the current implementation is not fast enough to be used in real-time applications, the nature of the voxel carving process described in this paper allows for the input point-cloud to be streaming as surface reconstruction occurs, since only the most recently captured points are needed at any given time. The extension to streaming input would allow for surface visualization during data acquisition.

References

[1] A. Adan and D. Huber. 3d reconstruction of interior wall surfaces under occlusion and clutter. *3DIMPVT*, pages 275–281, May 2011. 2

[2] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. *Proceedings of the Sixth Symposium on Solid Modeling*, pages 249–260, 2001. 1, 2

[3] J. A. Baerentzen. Octree-based volume sculpting. *IEEE Visualization*, 1998. 1, 2, 3

[4] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. Multilevel streaming for out-of-core surface reconstruction. *SGP*, pages 69–78, 2007. 2

[5] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. Parallel poisson surface reconstruction. *ISVC*, pages 678–689, 2009. 2

[6] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. *CVPR*, 2010. 2

[7] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor. Indoor localization algorithms for a human-operated backpack system. *3D Data Processing, Visualization, and Transmission*, May 2010. 3

[8] K. Denker, B. Lehner, and G. Umlauf. Real-time triangulation of point streams. *Engineering with Computers*, 27:67–80, 2011. 2

[9] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *SIGGRAPH*, pages 209–216, 1997. 2

[10] C. Holenstein, R. Zlot, and M. Bosse. Watertight surface reconstruction of caves from 3d laser data. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2011. 1, 2, 3, 6, 7

[11] H. Hoppe. Progressive meshes. *Computers and Graphics*, 1998. 1, 2

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Proceedings of SIGGRAPH'92*, pages 71–78, 1992. 2

[13] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink. Streaming computation of delaunay triangulations. *Proceedings of SIGGRAPH'06*, pages 1049–1056, July 2006. 2

[14] I. T. Jolliffe. *Principal Components Analysis, Second Edition*. Springer, 1986. 5

[15] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *SIGGRAPH*, 2002. 2

[16] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*, 2006. 1, 2

[17] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Grpahics - Proceedings of ACM SIGGRAPH*, 31(6), November 2012. 2

[18] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. *Symposium on Goemtry Processing*, pages 11–21, July 2004. 1, 2

[19] J. Kua, N. Corso, and A. Zakhor. Automatic loop closure detection using multiple cameras for 3d indoor localization. *IS&T/SPIE Electronic Imaging*, January 2012. 3

[20] F. Labelle and J. R. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, August 2007. 2, 5

[21] L. liang Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH Asia*, 31(137), November 2012. 2

[22] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4), July 1987. 2

[23] V. Sanchez and A. Zakhor. Planar 3d modeling of building interiors from point cloud data. *ICIP*, September 2012. 2

[24] T. Shao, W. Xu, K. Zhou, J. Wang, and D. L. B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Grpahics - Proceedings of ACM SIGGRAPH*, 31(6), November 2012. 2

[25] A. Sharf, D. A. Alcantara, T. Lewiner, C. Greif, A. Sheffer, N. Amenta, and D. Cohen-Or. Space-time surface reconstruction using incompressible flow. *ACM Transactions on Graphics*, 27(110), December 2008. 2

[26] S. A. A. Shukor, K. W. Young, and E. J. Rushforth. 3d modeling of indoor surfaces with occlusion and clutter. *International Conference on Mechatronics*, pages 282–287, April 2011. 2

[27] M. Smith, I. Posner, and P. Newman. Adaptive compression for 3d laser data. *The International Journal of Robotics Research*, 30(7):914–935, June 2011. 3

[28] G. Windreich, N. Kiryati, and G. Lohmann. Voxel-based surface area estimation: From theory to practice. *Pattern Recognition*, 26:2531–2541, 2003. 2

[29] J. Xiao and Y. Furukawa. Reconstructing the world's museums. *EECV 2012 Lectures in Computer Science*, 7572:668–681, 2012. 2

[30] Y.-K. Yang, J. Lee, S.-K. Kim, and C.-H. Kim. Adaptive space carving with texture mapping. *ICCSA*, 3482:1129–1138, 2005. 2, 3

[31] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195:942–960, 2006. 2

[32] K. Zhou, M. Gong, and B. Guo. Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):669–681, May 2011. 2