Fast, Automated, Scalable Generation of Textured 3D Models of Indoor Environments

Eric Turner, *Student Member, IEEE*, Peter Cheng, and Avideh Zakhor, *Fellow, IEEE*

***Abstract*—3D modeling of building architecture from mobile scanning is a rapidly advancing field. These models are used in virtual reality, gaming, navigation, and simulation applications. State-of-the-art scanning produces accurate point-clouds of building interiors containing hundreds of millions of points. This paper presents several scalable surface reconstruction techniques to generate watertight meshes that preserve sharp features in the geometry common to buildings. Our techniques can automatically produce high-resolution meshes that preserve the fine detail of the environment by performing a ray-carving volumetric approach to surface reconstruction. We present methods to automatically generate 2D floor plans of scanned building environments by detecting walls and room separations. These floor plans can be used to generate simplified 3D meshes that remove furniture and other temporary objects. We propose a method to texture-map these models from captured camera imagery to produce photo-realistic models. We apply these techniques to several data sets of building interiors, including multi-story datasets.**

***Index Terms*—Architecture, Floor Plan, Surface Reconstruction, LiDAR, Texture Mapping**

I. Introduction

LASER scanning technology is becoming a vital component of building construction and maintenance. During building construction, laser scanning can be used to record the as-built locations of HVAC and plumbing systems before drywall is installed. In existing buildings, blueprints are often outdated or missing, especially after several remodelings. Such scans can be used to generate building models describing the current architecture. Meshed triangulations allow for the efficient representation of the scanned geometry. In addition to being useful in the fields of architecture, civil engineering, and construction, these models can be directly applied to virtual walk-throughs of environments, gaming entertainment, augmented reality, indoor navigation, and energy simulation analysis. These applications rely on the accuracy of a model as well as its compact representation.

Generating an accurate model of indoor environments is an emerging challenge in the fields of architecture and construction for the purposes of verifying as-built compliance to engineering designs [1], [2]. This task is made more challenging by the GPS-deprived nature of indoor environments [3]. Another application that requires an exported mesh to retain as much detail as possible is historical preservation via a virtual reality model [4], [5]. Alternatively, building energy efficiency simulations can use watertight meshes of the environment to estimate airflow and heat distribution [6]. These simulations require simplified meshes as input, since finite element models are difficult to scale. It is also important to be able to generate an immersive visualization and walk-through of the environment for these applications, so experts can remotely inspect the scanned environment via telepresence, a task that currently requires expensive travel and on-site visits. Different applications require models of different complexities, both with and without furniture geometry. The modeling approaches detailed in this paper are useful for both types of applications, as shown in the examples in Fig. 1. Fig. 1a is a photograph of the scanned area: the hallways of an academic building, encompassing about 1,000 square meters of scanned area. Fig. 1d represents the captured 3D point-cloud of this area. Figs. 1b and 1c show a high-detail 3D mesh of 2.7 million triangles generated using the algorithm in Sec. III, with and without texturing, respectively. Figs. 1e and 1f show a low detail model of 2,644 triangles generated using the approach in Sec. IV, with and without texturing.

In this paper, we focus on ambulatory scanning platforms, where the sensor suite is carried by a human operator as the operator moves through the building environment [4], [7], [8]. These systems allow for rapid data acquisition and can be actively scanning for several hours at a time. They use 2D LiDAR scanners due to the cost and weight of full 3D laser range finders. The captured scans are used both to reconstruct the geometry of the environment and to localize the system in the environment over time. The datasets shown in this paper were generated by a backpack-mounted system that uses 2D LiDAR scanners to estimate the 3D path of the system over time as well as multiple scanners to generate geometry for the environment [9]–[12]. This system also has multiple cameras collecting imagery during the data acquisition process, which allows for scanned points to be colored or for generated meshes to be textured with realistic imagery.

In order to reconstruct the observed geometry, the position and orientation of the scanning system must be estimated for each time-instant during the acquisition. To localize the datasets presented in this paper, we matched successive 2D scans of a horizontally-oriented scanner in order to estimate the change in 2D position and orientation of the system between scans [12]. These incremental changes in pose are refined by a global graph optimization step that constrains the system path whenever an area is revisited [13]. These revisits

E. Turner and A. Zakhor are with University of California Berkeley Department of Electrical Engineering and Computer Sciences Berkeley, CA 94720 USA (email: elturner@eecs.berkeley.edu; petercheng00@gmail.com; avz@eecs.berkeley.edu)

This research was conducted with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

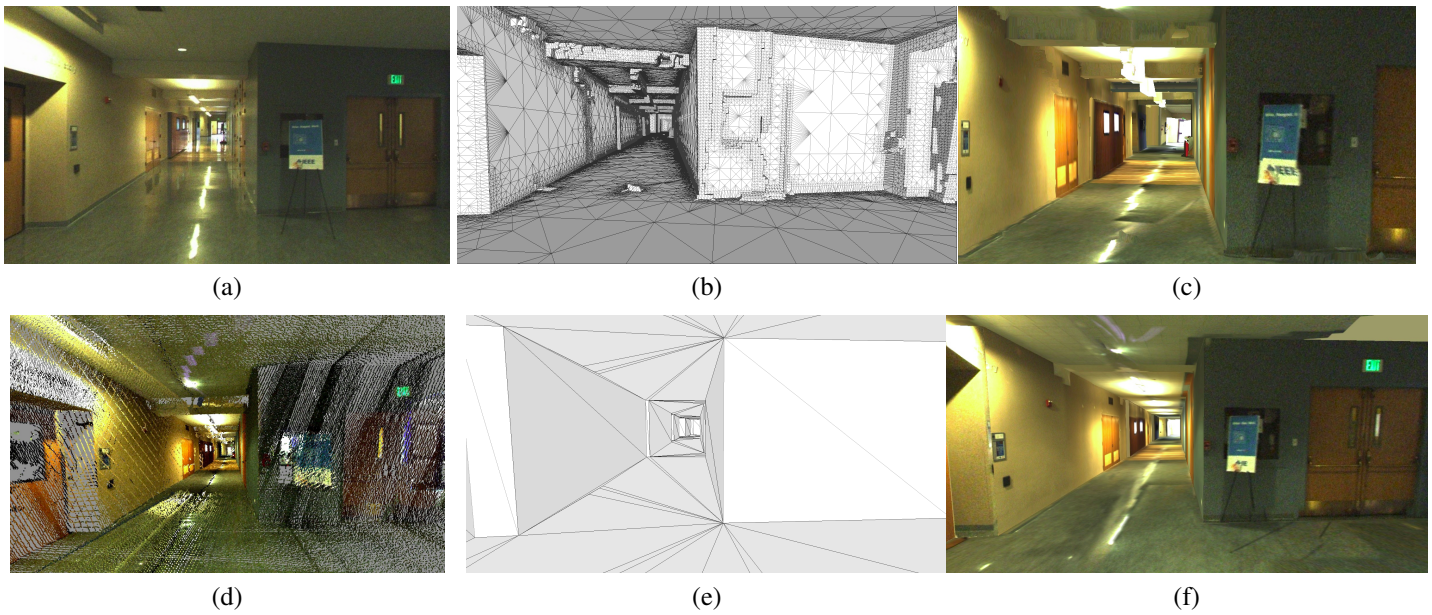


Fig. 1. Models generated with the techniques described in this paper: (a) photograph of scanned area, academic building; (b) surface carving model of this area; (c) surface carving model with texturing; (d) point cloud of captured scans; (e) extruded floor plan model of area; (f) extruded floor plan with texturing.

are automatically detected and constrained in order to produce an accurate 2D trajectory of the system. The elevation of the system is computed using a similar technique with a vertically mounted scanner [10].

Once the path of the system over time is recovered using the above localization schemes, the pose of each sensor is known at any given time, where pose refers to 3D position and orientation. A point cloud of the scanned environment can be generated by performing a rigid translation and rotation of scanned points from the sensor's coordinate frame at each timestamp to the world coordinate frame. These points can then be back-projected to the image plane of the temporally nearest camera image in order to assign color information [10]. Fig. 1d shows an example of a captured point cloud that is colored with imagery in this manner.

In this paper, we show two surface reconstruction techniques for 3D point clouds generated by ambulatory systems. Sec. III describes a method that preserves these fine details while remaining robust to registration errors and noise from the input scans, which generates models such as the one shown in Fig. 1b. Sec. IV describes a method that first generates a floor plan of the building environment, then creates a 2.5D model by extruding the floor plan vertically using captured height information. Such a model represents the floors, walls, and ceilings of a model efficiently, while removing geometry associated with furniture and other objects within the building. This type of model is shown in Fig. 1e. The output of both of these modeling techniques can be texture-mapped with captured camera imagery, as described in Sec. V. An example of texture-mapping these two modeling processes is shown in Figs. 1c and 1f. Lastly, we will show the results of all of these techniques on a variety of data sets in Sec. VI.

II. Related Work

Traditional industry standard building scanning is to use static scanners. Such scanners are mounted on tripods, and moved from area to area in the building [14]–[18]. This scanning process is labor intensive and slow, but results in highly accurate point clouds after stitching. In order to automate indoor scanning, many mobile systems have been introduced. Wheeled platforms that carry scanning equipment and are manually pushed through the environment are popular [5], [19]. Mobility of such systems is limited, since they are unable to traverse rough terrain or stairs easily. Others have investigated mounting laser range finders on unmanned aerial vehicles [20], [21]. Such platforms are agile in that they can scan difficult-to-reach areas. Such unmanned platforms are limited by short battery life and cannot scan for long durations. This paper focuses on scans from mobile systems. Such systems allow for faster acquisition of the data, but also introduce increased mis-registration error.

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 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 [5], [22]–[26]. In addition to large flat regions, building interiors also contain many small details, such as furniture. A surface reconstruction scheme must be able to represent the large surfaces in a building with an efficient number of elements and preserve their sharp features. The fine details of furniture are useful for some applications whereas others require furniture to be removed.

Mobile mapping systems use range scanners to create

a dense 3D point-cloud representation of the environment geometry [7], [11], which can be used to develop full 3D models [5], [27]. One 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 [28] or furniture [29]–[31]. Unfortunately, these techniques are heavily dependant on the variance of the database of shapes available and are prone to errors due to mislabeling. Many surface reconstruction techniques applied to building architecture 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 [17], [18], [28]. 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 [32]. Existing techniques that attempt to preserve fine detail for architecture models often require computationally expensive global optimizations [27]. Those approaches work well for a limited modeling environment, but do not scale well. The largest tested model in [27] consists of 3.3 million points, whereas the techniques described in this paper are easily applied to models with 115 million points [33]. There are also many Kinect-based approaches [34]–[36], which offer high-detailed models, but generate dense enough models that prevent scalability to building-level scanning. One advantage of Kinect-based systems is the large amount of data produced in each frame, which allows for accurate models based on averaging techniques. However, the additive noise in each Kinect scan is much larger than other scanning technologies, such as time-of-flight, thus requiring aggressive filtering to be used with any Kinect-based point-cloud. It is desirable to develop techniques that (a) use a volumetric approach to ensure watertightness, (b) preserve sharp, planar features as well as fine detail, and (c) are fast and memory efficient even with large models.

There have been several algorithms that reconstruct surfaces from point-clouds using a volumetric approach via partitioning Delaunay Tetrahedralizations generated from input point clouds [22], [37]. The downsides to such techniques are that (a) the complexity of the output surface scales with number of input points, (b) they break down under noise due to scan misregistration, (c) they are optimized for smooth and continuous surfaces, and (d) they require a global optimization step. These factors limit the scalability of existing approaches to mobile scanning of indoor environments. While advancements have been made to perform these computations in an efficient and out-of-core manner [38], [39], the resulting models are too large to be practical for graphical or simulation applications.

Implicit surface reconstruction techniques generate watertight meshes and can be applied to large models using distributed computing techniques [25], [40]–[42]. These techniques are unsuitable for modeling man-made architecture, since output models lack sharp features due to implicit surfacing from Gaussian basis functions. Additionally, many common triangulation schemes for implicit surfaces result in uniform elements [43], [44], which are undesirable for large, flat surfaces that can be modeled just as accurately with fewer elements. Such techniques also often require mesh smoothing, further reducing accuracy [5]. Algorithms that adaptively mesh an isosurface or simplify an existing mesh rely on the local feature size of a model [24], [45]–[47]. Models with flat regions or sharp corners, where the curvature approaches zero or infinity, can become degenerate or have poor quality. 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 [5], [23], [48]–[51]. Such approaches are robust to noise and registration errors, but challenges with voxel representations are memory and computational intensity.

A number of simplified building modeling algorithms have been developed, most of which assume vertical walls, rectified rooms, and axis-alignment [32], [52]. Being able to make assumptions about the planarity of an environment has been used successfully to model only the major features of scanned objects even in the presence of high noise [53]. A simplified model tends to be more robust to noise and clutter and allows for faster processing of data. One of the major limitations of these techniques is that they are typically developed only for axis-aligned models or fundamentally change the topology of minor areas, such as ignoring doorways, shapes of rooms, or small rooms entirely. Our approach described in Sec. IV generates a 2D floor plan of the building, then uses wall height information to generate a 3D extrusion of this floor plan. Such blueprint-to-model techniques have been well-studied [54], [55], but rely on the original building blueprints as input. Prior work on automatic floor plan generation use dense 3D point-clouds as input, and take advantage of the verticality of walls to perform histogram analysis to sample wall position estimates [26], [56], which are in the same format as a grid map for particle filtering [57]. In situations where dense 3D point-clouds are available, we apply similar techniques to recover point estimates of wall positions.

When generating a mesh of indoor environments, one application allows for imagery to be used to create texture-maps for the mesh. Each surface of the environment is represented by a generated texture image. There are many existing approaches to stitching together multiple images to produce a larger, seamless image [58]–[63]. Generally, parts of images are matched to each other by detecting feature points and identifying matches. Images are then transformed to maximally align matches. There exist impressive techniques that perform high-quality texture-mapping given sufficient coverage of an object with high-calibrated scanners [36], but the process of using noisy camera orientations with fast-moving scanners is still a challenge. Feature matching works best when unique visual references exist in the environment that can be detected in multiple images. Unfortunately, many indoor environments have a high prevalence of bare surfaces as well as repeating textures, such as windows and doors.

III. Detailed Mesh Reconstruction

Many practical applications require the captured geometry of building interiors to preserve as much detail as possible. This detail includes all static objects in the scene, such as furniture or temporary items. The surface reconstruction method we propose in this section generates a watertight mesh that preserves all details in the original scan points [33]. The approach described below is a modification of voxel carving to address these issues. We also introduce memory-efficient data structures that produce models that preserve fine details with an efficient number of elements. In subsection III-A, we detail the voxel carving step, which produces a volumetric representation of the scanned environment. In subsection III-B, we describe our approach to generate planar surfaces in order to form a watertight mesh of the captured volume.

A. Voxel Carving

We partition space volumetrically into interior and exterior sets to ensure the boundary between these areas is watertight. This *interior* and *exterior* volume classification is performed on a voxel grid. Initially, all voxels are assumed to be *exterior*, referring to any space that never had a line-of-sight with the scanners. This label applies to the interior volume of solid objects, as well as the area completely outside the building environment. 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 a scanner to a corresponding scan point. If a laser passes through a voxel, that voxel is considered *interior*.

Additionally, the sweeping nature of the laser scanning process is utilized by also carving voxels that reside between two adjacent scan-lines [5], [33]. As the scanning system moves from pose t_i to pose t_{i+1}, the scan-lines $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j}$, and $p_{i+1,j+1}$ are captured, as shown in Fig. 2a. These scan-lines are bilinearly interpolated, as shown in Fig. 2b, so that all intersected voxels are carved, which is depicted in Fig. 2c.

In most common voxel representations, 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 [5], [34]. This step adds redundant computation and storage overhead. Rather than storing all relevant voxels in memory, 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.

B. Planar Surface Meshing

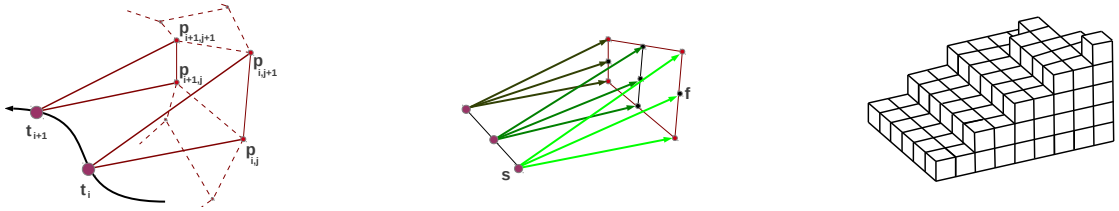
Our procedure for surface reconstruction of voxels can be broken into two parts. First, estimates of planar regions are found around the boundary faces of these voxels. These regions are formed from connected sets of voxel faces, all of which are positioned on best-fit planes. Second, each region is 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.

We wish to encourage the output surface to contain large planar regions. Such regions accurately model most man-made structures and the dominant surfaces in a building environment: the floors, walls, and ceiling. As shown in Sec. V, surfaces composed of large planar regions have improved aesthetic quality after texture-mapping is applied. 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 [64]. Adjacent regions of voxel faces are progressively merged by attempting to model their union with a single best-fit plane. 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.

Once the set of voxel faces has been partitioned into planar regions, it is necessary to triangulate these regions. Taking advantage of the existing voxel grid ensures 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 [46]. An example region of voxel faces is shown in Fig. 3a. 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 Fig. 3a. 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 Fig. 3b. The tree is triangulated by placing vertices at the center and corners of the leaf nodes, as shown in Fig. 3c. 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.

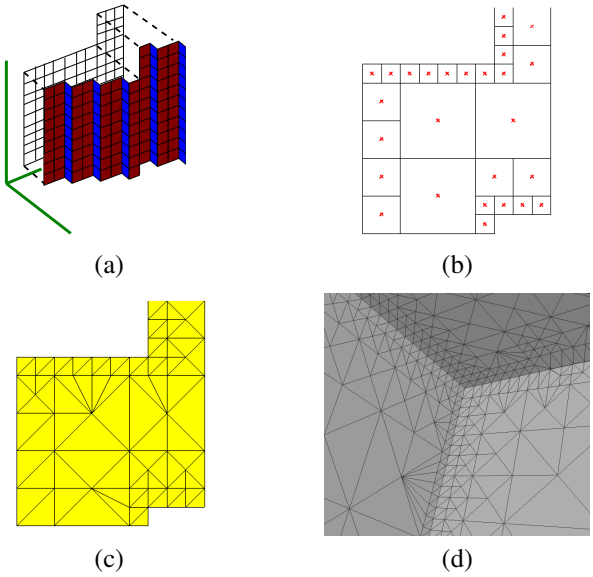
To ensure that the borders between planar regions are represented sharply, the vertices that are shared by multiple regions are snapped onto the intersection of those regions. This step yields a watertight mesh across regions, as can be seen in the intersection of three regions at the corner of a room in Fig. 3d.

As shown in Sec. VI, this approach results in models that



(a) (b) (c)

Fig. 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) (d)

Fig. 3. (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.

preserve geometric detail up to the voxel resolution. Improving the input resolution allows for finer detail to be represented in the output model, at the cost of increased run-time.

IV. Simplified Mesh Reconstruction and Floor Plan Generation

While the technique described in Sec. III has many applications, it is often necessary to generate models rapidly or with very few elements, that do not represent objects such as furniture in the environment. Such models are especially useful for building simulation applications, which are restrictive in the number of elements used to represent building geometry [6]. In this section, we describe a surface reconstruction technique that first constructs a 2D floor plan of the area for each level of the building, then extrudes these floor plans into 3D models using estimates of the floor and ceiling heights in each room scanned. We refer to this process as 2.5D modeling [65].

A novel contribution of our method is the use of room labeling to enhance building models. One motivation for existing work has been to capture line-of-sight information for fast rendering of building environments [66], whereas others have partitioned environments into segments for efficient localization and tracking [67]. These approaches are meant to create easily recognizable subsections of the environment, whereas our proposed room labeling technique uses geometric features to capture semantic room definitions for both architectural and building energy simulation applications.

A. Wall Sampling

The input data used during floor plan generation consist of points in the (x,y) horizontal plane, which we call wall samples. These points depict locations of walls or vertical objects in the environment. We assume that interior environments satisfy "2.5-Dimensional" geometry: all walls are vertically aligned, whereas floors and ceilings are perfectly horizontal. Many mapping systems use a horizontal LiDAR scanner to estimate a map of the area as a set of wall sample positions in order to refine estimates for scanner poses [8], [12]. Such 2D wall samples are often generated during the localization process and do not require separate processing to produce. These mobile mapping systems often have additional sensors capable of estimating floor and ceiling heights at each pose [10], [20]. The input to our algorithm is a set of 2D wall samples, where each sample is associated with the scanner pose that observed it, as well as estimates of the floor and ceiling heights at the wall sample location.

An alternate method of computing wall samples is to subsample a full 3D point-cloud to a set of representative 2D points [26], [56], [65]. This process cannot be done in a streaming fashion, since it requires a complete point cloud as input, but it can provide more accurate estimates for wall positions than a real-time particle filter. Such an approach is useful when representing dense, highly complex point clouds with simple geometry. Under the 2.5D assumption of the environment, wall samples can be detected by projecting 3D points onto the horizontal plane. Horizontal areas with a high density of projected points are likely to correspond to vertical surfaces. This approach works well to capture permanent wall features and ignore furniture and other interior clutter.

This approach creates models by generating a floor plan separately for each level in the building. Some localization systems that rely on 2D grid maps of wall samples are capable of detecting when the operator moves from one level to another and automatically partition their output grid maps accordingly [8]. If the wall samples are generated from point-

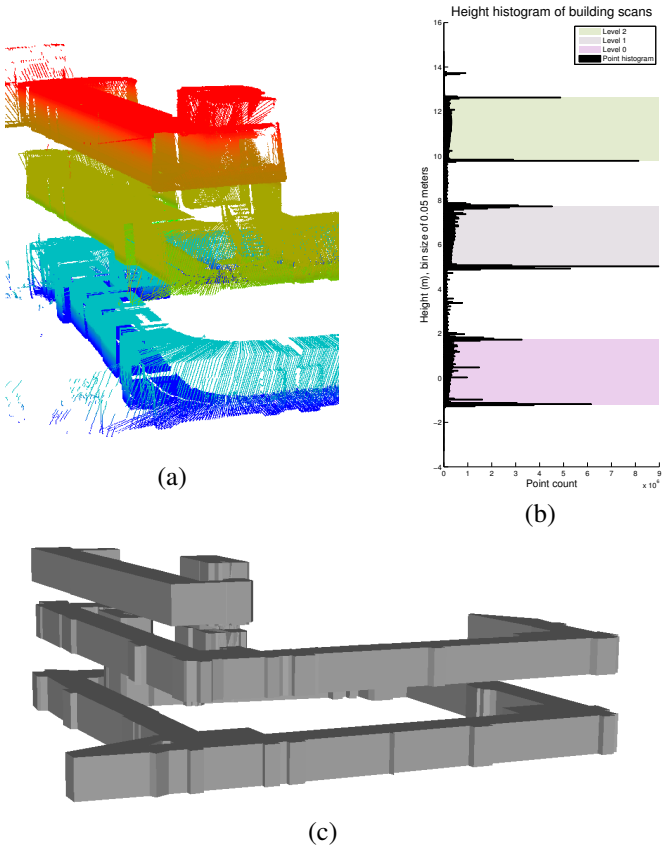


Fig. 4. An example point-cloud partitioning by height: (a) the input point-cloud, showing geometry for three levels; (b) the vertical histogram showing estimates of each building level height; (c) the produced mesh of this building scan.

clouds, then a histogram approach can be used to separate the point-cloud by levels [26]. Fig. 4a shows an example point-cloud, colored by height, which contains multiple levels. By computing a histogram along the vertical-axis of the point-cloud, it is possible to find heights with high point density, which indicates the presence of a large horizontal surface. An example of this process is shown in Fig. 4b, where peaks in the histogram correspond to the floors and ceilings of each scanned level in the building. Points scanned from above, in a downward direction, are used to populate a histogram to estimate the position of each floor, and the histogram used to estimate the position of each ceiling is populated by points scanned from below. The local maxima of these two histograms show locations of likely candidates for floor and ceiling positions, which are used to estimate the number of scanned levels and the vertical extent of each level. Fig. 4c shows the final extruded mesh with all three scanned levels.

B. Floor Plan Generation

For each scanned level of the building, we generate a 2D floor plan by partitioning space into *interior* and *exterior* domains, in the same manner as in Sec. III. The boundary between these two domains are exported as the building walls in the floor plan.

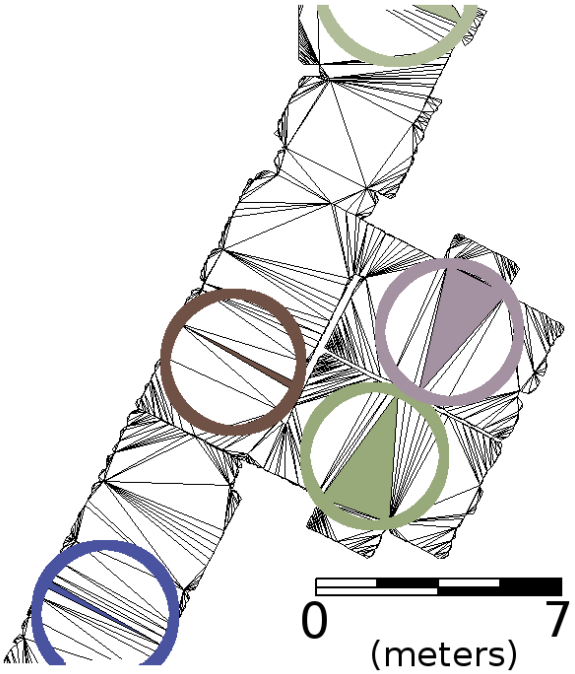
The input wall samples are used to define a volumetric representation by generating a Delaunay Triangulation on the plane. Each triangle is labeled either interior or exterior by analyzing the line-of-sight information of each wall sample. Initially, all triangles are considered exterior. For every scanner position over time, the line segments from the scanner's position to each associated wall sample are considered. If a triangle is intersected by one of these line segments, then it must be interior, since the scan was not occluded by any solid objects. Each such intersected triangle is relabeled to be interior. This process is similar to the carving process described in Sec. III, but traces 2D lines across triangles, rather than 3D lines across voxels. The line-of-sight information is analyzed from each pose of the system. The subset of triangles that are intersected by these projected laser scans are considered interior, while the remaining triangles are denoted as exterior and discarded.

C. Room Labeling

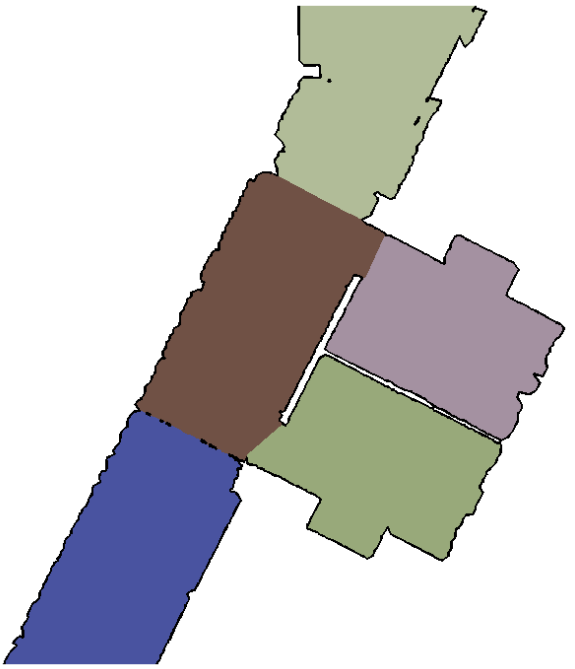
Once we have a volumetric partitioning that defines the interior space of a building, we can use this information to model individual rooms inside the building. We define a *room* to be a connected subset of the interior triangles in the building model. Detected rooms should match with real-world architecture, where separations between labeled rooms are located at doorways in the building. We model room labeling as a graph-cut problem. First, a rough estimate for the number of rooms and a seed triangle for each room is computed. A seed triangle is representative of a room, where every room to be modeled has one seed triangle. These seeds are used to partition the remainder of interior triangles into rooms. This process typically over-estimates the number of rooms, so prior knowledge of architectural compliance standards is used to evaluate each estimated room geometry. Using this analysis, the number of ill-formed rooms is reduced, providing an update on the original seed points. This process is repeated until the set of room seeds converges.

We use the Delaunay property of the triangulation to identify likely seed triangle locations for room labels. It is unlikely that the circumcircles of the interior triangles intersect the boundary walls of the carved floor plan, which causes these circles to overlap only interior area. Each triangle's circumradius provides an estimate of the local feature size at its location on the floor plan boundary polygon. An example of this process is shown in Fig. 5, with the highlighted triangles in Fig. 5a showing the chosen seed locations. Triangles with larger circumradii are likely to be more representative of their rooms than those with smaller circumradii. We form the initial set of room seeds by finding all triangles whose circumcircles are local maxima. That is, a seed triangle will have a circumcircle with a larger radius than any other circumcircle that intersects it. This process selects the largest triangles that encompass the space of rooms as the seeds for room labeling. Fig. 5b shows example seed triangles and their corresponding circumcircles. The result is an estimate of the number of rooms and a rough location for each room.

All interior triangles in the floor plan are then associated with one of these seeds, to form the total area of each room. This step can be performed as a graph-cut on the dual of



(a)



(b)

Fig. 5. Example room seed partitioning on an interior triangulation: (a) the room seed triangles, and their corresponding circumcircles; (b) room labels propagated to all other triangles.

the triangulation. Specifically, each triangle is a node in the graph, with the edge weight between two abutting triangles as the length of their shared side. Performing a min-cut on this graph partitions rooms to minimize inter-room boundary length. In other words, rooms are defined to minimize the size of doors. This process propagates the room labels to every triangle, and the boundaries between rooms are composed of

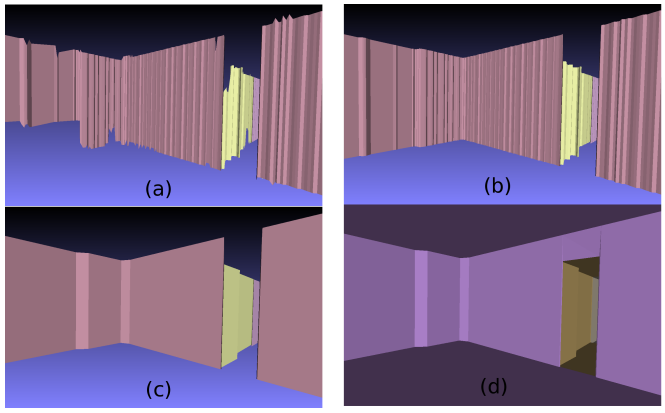


Fig. 6. Example of creating a 3D extruded mesh from 2D wall samples: (a) walls of generated floor plan with estimated height ranges; (b) floor and ceiling heights are grouped by room; (c) simplification performed on walls; (d) floor and ceiling triangles added to create a watertight mesh.

only the smallest edges in the triangulation. The result of this process is shown in Fig. 5c.

The initial room seeds typically over-estimate the number of rooms, since a room may have multiple local maxima. This case is especially true for long hallways, where the assumption that one triangle dominates the area of the room is invalid. The solution is to selectively remove room seeds and redefine the partition. A room is considered a candidate for merging if it shares a large perimeter with another room. Ideally, two rooms sharing a border too large to be a door should be considered the same room. By Americans with Disabilities Act Compliance Standards, a swinging door cannot exceed 48 inches in width [68]. Accounting for the possibility of double-doors, we use a threshold of 2.44 meters, or 96 inches, when considering boundaries between rooms. If two rooms share a border greater than this threshold, then the seed triangle with the smaller circumradius is discarded. With a reduced set of room seeds, existing room labels are discarded and the process of room partitioning is repeated. This iteration repeats until the room labeling converges.

D. 2.5D Model Extrusion and Simplification

Partitioning the floor plan model into separate rooms is useful for both model refinement and 3D mesh extrusion. In order to generate a 3D mesh, we extrude the floor plan vertically using estimates of the floor and ceiling height at each wall sample location. These heights estimates are often very noisy, due to clutter in the room, or the sparsity of samples. Fig. 6a shows an example room with these initial height estimates. The height estimates of samples within each room are combined to form a single floor height and a single ceiling height. By taking the median floor and ceiling heights in each room, the resulting model still captures accurate geometry and variations in ceiling heights across the building, but ensures that each room is modeled simply and sharply. The result of this median filtering is shown in Fig. 6b.

In many applications, it is useful to reduce the complexity of the floor plan representation, so that each wall is represented by a single line segment. This step is often desirable in order

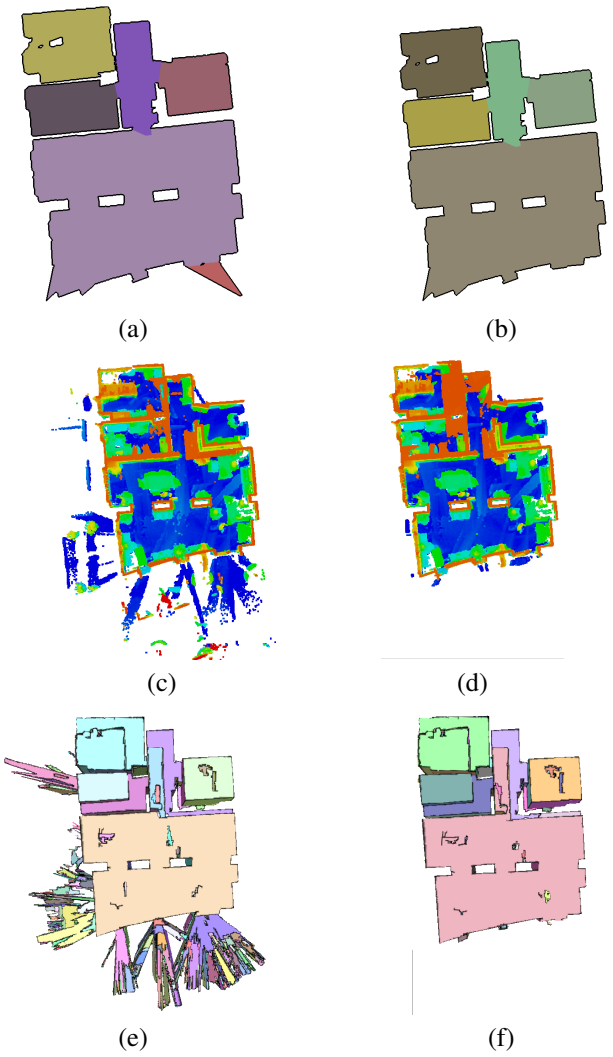


Fig. 7. Using room labels to trim and improve models. Top-down view of: (a) Floor plan before room trimming; (b) floor plan after trimming; (c) point cloud before trimming; (d) point cloud after trimming; (e) surface carving before trimming; (f) surface carving after trimming.

to attenuate noise in the input wall samples or to classify the walls of a room for application-specific purposes. We simplify wall segments using a variant of QEM [45], [65]. Performing this step after room partitioning allows for the fine details in the doorways between rooms to be preserved, but the large surfaces within a room to be simplified more aggressively. Fig. 6c shows the results of this simplification process. The floor and ceiling mesh is is taken from the simplified 2D triangulation of the floor plan, as shown in Fig. 6d.

E. Model Refinement using Room Labels

Room labeling within a model provides an effective mechanism to prevent misrepresentation of poorly scanned areas. The mobile scanning system does not necessarily traverse every room and may only take superficial scans of room geometry while passing by a room's open doorway. When a room is not entered, the model is unlikely to capture sufficient geometry and therefore it is necessary to remove this poorly scanned area from the model. If none of the triangles for a room within the floor plan are intersected by the scanner's path, we can infer the room is never entered. The room's triangles are then relabeled from interior to exterior, removing it from the floor plan. Figs. 7a and 7b show an example floor plan before and after such trimming occurs, respectively. Note the removal of a sharp extrusion in the bottom-right corner, which was a partial scan through a window.

Similar refinement can be used to improve the 3D carving method described in Sec. III. Due to the nature of voxel carving, laser scans that pass through a building window or open doorway can capture geometry outside of the desired scanned area. Since the outside volume is only observed from a few angles, the resulting carving produces undesirable artifacts. Using the 2.5D extruded floor plan, we can automatically remove scans from the input point cloud that fall outside our desired area. Figs. 7c and 7d show the corresponding point clouds before and after the result of this trimming, colored by height with the ceiling points removed. Any scans that pass through windows or doorways are removed. As a result, the surface carving of these point clouds, as shown in Figs. 7e and 7f respectively, can be improved by reducing these undesirable artifacts.

V. Texture Mapping

Texture-mapping virtual models provides enhanced realism for visualization purposes and allows for finer detail to be represented in the final product. A main challenge of texture mapping models from mobile scanning platforms is the presence of increased noise and uncertainty in camera poses. As a result, common texture mapping methods produce poor results. One of the most prominent challenges is to ensure texture continuity along large uniform areas, while allowing texture boundaries to fall along natural geometric boundaries [69]. The surface reconstruction methods described in this paper are designed to aid in minimizing the visibility of discontinuities in applied texture. Each region in the geometric model is textured independently. For the surface carved models discussed in Sec. III, these planar regions are fit explicitly to the voxel carving. For the extruded floor plan models described in Sec. IV, these planar regions are each extruded wall, as well as the floor and ceiling surfaces from each room.

Our datasets often contain long 1-dimensional chains of images, such as in Fig. 8, which frequently lead to error accumulation in the form of translational drift. Fig. 8a depicts an integration of image stitching with the iterative global localization algorithm applied to estimate the movement of the mobile system [9]. Fig. 8b depicts an output image from AutoStitch, which is software based on research in the related area of panorama generation [70], [71]. Both methods were thoroughly tuned for the shown example, but due to the factors mentioned in the previous paragraph, they still show significant amounts of drift and distortion, as well as loss of alignment to the environment geometry. Fig. 8c shows the output of the method described in this paper [69].

The overall block diagram for the proposed texture-mapping procedure is shown in Fig. 9. The details of image selection and alignment are described in Subsect. V-A. The techniques

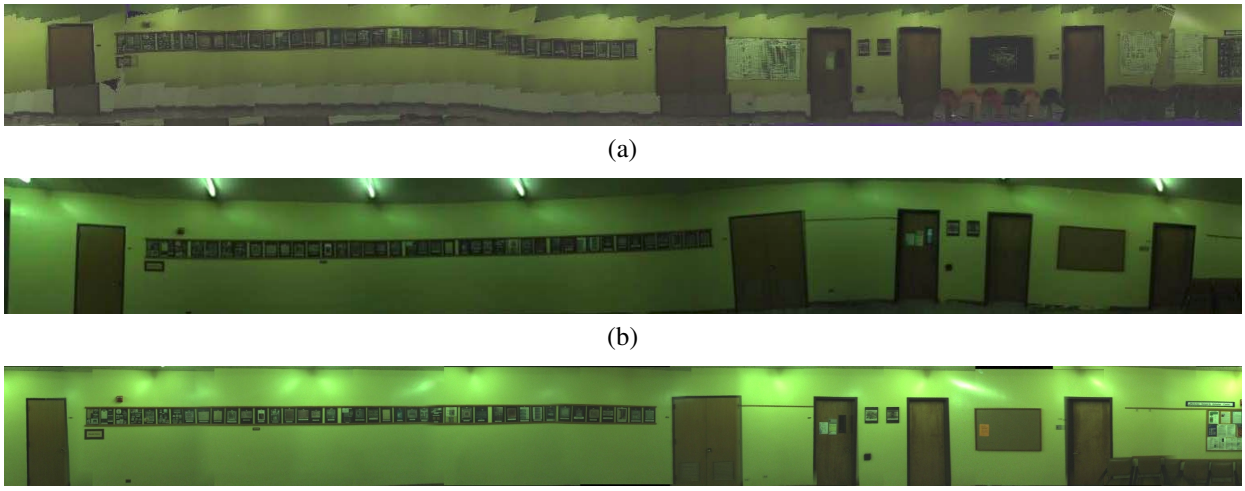


Fig. 8. Texture alignment via (a) the graph-based localization refinement algorithm; (b) the AutoStitch software package; (c) the method proposed in this paper.

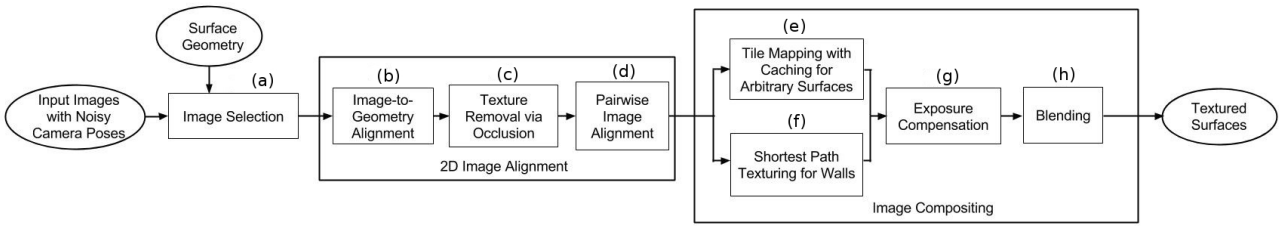


Fig. 9. The proposed texture-mapping procedure

used for image compositing and generating the final texture maps for each surface are described in Subsect. V-B.

A. Image Selection and Alignment

The cameras on the mobile scanning hardware typically take several images a second for the entire duration of the data acquisition process. This capture results in thousands of potential images to be used to texture any given surface in the 3D geometry. Our objective when texture-mapping each surface is to determine which images will contribute to the surface's texture. Each of those selected images is projected onto the surface and blended together to form the surface's final texture.

To begin generating a texture, we first obtain a suitable set of images for each surface, which together form a desirable set of candidate images for use in texturing that surface. This selection process is depicted in box (a) of Fig. 9. These images are selected in order to satisfy three criteria respective to their surface. First, each selected image must have unoccluded line-of-sight from its camera location to the surface, which can be verified with ray-polygon intersection tests using the surface geometry [72]. Second, the set of images selected for a surface must together contain imagery that spans the entirety of the surface. Third, images should be selected such that all areas on the target surface have at least one image with a highly optimal viewing angle to it. These three criteria are satisfied by discretizing each surface into small rectangular tiles. All images with a clear line-of-sight to each surface tile are considered, and out of that set, the image with the best viewing angle is selected. The process of removing images whose lines-of-sight to the surface are occluded is depicted in box (c) of Fig. 9. Optimal viewing angle can be objectively defined by maximizing the scoring function $\frac{1}{d}(-1 \cdot \vec{c}) \cdot \vec{n}$ as shown in Fig. 10, where d is the distance between the camera center and position on region surface, $\vec{n}$ is the surface's normal vector, and $\vec{c}$ is the camera's look axis [69]. By collecting the optimal image selected for each tile on a surface, we produce the set of candidate images that will be used for texturing that surface. Once this set of candidate images is selected for a surface, the images can be directly projected onto the surface's geometry using their noisy recovered camera poses, resulting in a complete texture. This projection process occurs in box (b) of Fig. 9. Due to noise in the estimates of camera positions, the image projections line up poorly, resulting in highly visible discontinuities at the boundaries between images.

The first mechanism used to align the projected images is to detect edges in images, and match them to their counterparts in surface geometry. Edges in images are found using Hough Transforms, whereas edges in geometry are the triangulation edges that occur between planar regions. For each edge detected in an image, potential matches in geometry are found within a distance and orientation threshold. Using a RANSAC

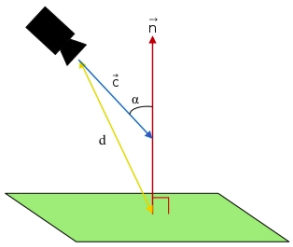


Fig. 10. Camera angle α and distance d are minimized by maximizing the scoring function $\frac{1}{d}(-1 \cdot \vec{c}) \cdot \vec{n}$

framework, a rigid transformation is calculated that maximally aligns as many of these potential matches as possible within each image [69], [73]–[75]. Applying these transformations to images ensures that each surface's texture consists only of the correct imagery for that surface, as boundaries of regions are usually clearly visible in images, and can be successfully aligned. This results in sharp, continuous borders at the boundaries of textures, which indirectly improves alignment between images both within a region and across different regions as well. If only one edge match is present, or all edge matches are parallel, then the transformation has one degree of freedom, which will be handled in the optimization step below.

To further refine image locations, we directly improve inter-image alignment by performing pairwise matching of SIFT features across all pairs of overlapping images on a surface. This step is depicted in box (d) of Fig. 9. For each pair of images, their SIFT matches are input into a RANSAC framework in order to compute a single transformation to robustly match feature points between each pair of overlapping images [73], [76].

In order to reconcile all image-geometry transformations as well as relative pairwise image transformations, a weighted linear least squares optimization problem is solved [69]. This step is depicted in box (e) of Fig. 9. An example setup $\min_{\vec{\beta}} ||W^{\frac{1}{2}}(A\vec{\beta} - \vec{\gamma})||_2^2$ with 3 images is as follows:

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -m_2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \vec{\beta} = \begin{pmatrix} x_1, \\ x_2, \\ x_3, \\ y_1, \\ y_2, \\ y_3 \end{pmatrix} \quad \vec{\gamma} = \begin{pmatrix} dx_{1,2}, \\ dy_{1,2}, \\ dx_{2,3}, \\ dy_{2,3}, \\ -m_2 gx_2 + gy_2, \\ gx_1, \\ gy_1, \\ tx_1, \\ ty_1, \\ tx_2, \\ ty_2, \\ tx_3, \\ ty_3 \end{pmatrix} \quad \vec{W} = \begin{pmatrix} 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01 \end{pmatrix} \tag{1}$$

The variables we wish to solve for are the x_i and y_i positions of images and the system of equations constrain the feature-based distances between pairs of images, images to geometry edges, and the original noisy camera poses. In Eq. 1, a feature-based displacement of $(dx_{1,2}, dy_{1,2})$ was calculated between images 1 and 2. This displacement corresponds to the first and second row of A, whereas the third and fourth row of A represent the same for images 2 and 3. Rows 5 through 7 correspond to the earlier geometry-based transformations. Specifically, row 5 corresponds to a geometry-based constraint of image 2's location to a line of slope m_2, passing through point (gx_2, gy_2), while rows 6 and 7 correspond to a fixed location for image 1 without any degrees of freedom. Rows 8 through 13 correspond to the original camera locations for each image (tx_i, ty_i).

The original camera poses are needed due to a potential lack of feature matches in all images or a lack of geometry alignment constraints to generate a non-degenerate solution. We assign the original noisy poses a weighting factor of 0.01, whereas all other equations are weighted at 1. Since this problem is linear, it can be solved efficiently. After applying these resulting positions, the alignment of the projection of all images onto a surface results is significantly improved, with any visible image transitions mostly due to brightness differences or more significant 3D geometry errors.

B. Image Compositing

Brightness differences in images occur because our cameras have variable exposure settings. This can result in significant brightness changes across adjacent images, particularly in areas near light sources. To adjust for these discontinuities in intensity, a relative gain is computed between each pair of overlapping images. Depicted as box (g) in Fig. 9, this relative gain is obtained by calculating the scaling factor between the average intensity of the region of pixels common to both images. These pair-wise relative gains are then used as observations in a least-squares optimization problem, whose solutions results in a single gain for each image that minimizes brightness differences between all adjacent images [69].

Now that images are well-aligned and their brightness has been equalized, the final step is to combine them into a single texture. Earlier, when each region's set of images was selected, a scoring function was used to determine the optimal image for each tiled area on a region. This scheme can be reused as a simple and efficient texturing method by texturing each tile with the image selected for it. This method can be further improved by employing a spatial cache and blending adjacent tiles in order to encourage adjacent tiles to share the same image where possible and reduce the visibility of seams otherwise. This step is indicated as box (h) in Fig. 9. This method can be applied to any type of surface geometry and successfully utilizes images taken from all arbitrary poses. Unfortunately, this approach tends to use a large number of images, which increases the probability that image seams will be visible in the final texture, due to imperfect matching.

In the context of indoor environments and side-facing cameras, a large number of surfaces are long and planar, with a high density of images taken side-by-side. This situation occurs for nearly all wall surfaces. As a result, selecting images for texturing is often a 1-dimensional problem. For these cases, rather than selecting images that are independently optimal for their own areas, we instead obtain a set of images with the goal of minimizing the visibility of image boundaries overall. The visibility of a boundary between a pair of images can be simply calculated as the sum of squared distances of pixel values in their overlapping area. Using these values as edge costs, and each image as a node, a shortest-path problem

can be constructed where the solution represents a set of images that spans the surface while still containing minimally visible image boundaries. This special case is depicted in box (f) of Fig. 9. These images are then mapped onto the surface and blended together to form a texture. This method gives preference to images that have a wide coverage, and encourages image boundaries to occur only where adjacent images are most well-aligned. This texturing method provides superior results, and is the preferred method for applicable surfaces.

VI. Results

In this section, we show example results of our techniques in Fig. 11, 12, 13, and 14. In doing so, we analyze the size of the produced models and associated run times, compare our results to state-of-the-art methods, and discuss limitations.

The resulting meshes of the methods described in Sec. III and IV are each useful for different applications. Fig. 14 shows the results of modeling the scans collected in a hotel lobby. The input point clouds for this model consisted of 70.4 million points, which comprised 5.16 GB on disk. The generated models cover 2,317 square meters, or about 25,000 square feet. The surface carving model, as shown in Figs. 14a and 14e, is represented by 2.65 million triangles. The extruded floor plan, as shown in Figs. 14c and 14g, is modeled with 2,944 triangles. This reduction by a factor of a thousand means that finer details in the model such as furniture or drop ceilings are not present, but can aid in many applications, including texture-mapping.

The size of the resulting texture-mapped models, using the method discussed in Sec. V, is much larger due to the generated high-resolution textures from camera imagery. The texture-mapping of the surface carving model, as shown in Figs. 14b and 14d, is 1.45 GB on disk comprised of textures for 1,277 distinct surfaces. The texture-mapping of the extruded floor plan depicted in Figs. 14f and 14h takes up 488 MB on disk with 566 surfaces. The largest surface for texturing is the floor of the main lobby area, whose texture is represented by a 14210×10555 image. A video fly-through of these models can be found online [77].

Run-time analysis was performed on the dataset shown in Fig. 12. The input to this dataset contains 25 million points. The code was run on a laptop with an Intel i7-2620M processor with 8 GB of RAM. All approaches presented in this paper were implemented in C++ as single-threaded programs. The voxel carving method described in Sec. III, at 5 cm resolution, took 55 minutes of processing. 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 [5]. Computation time was recorded for this same dataset with a resolution of 2 cm. Voxel carving took 12 hours and 10 minutes at this resolution and surface meshing took 9.5 minutes.

The same input point-cloud was processed using the 2.5D modeling approach described in Sec. IV on the same hardware. The processing step of extracting wall samples from the input point-cloud took 84.3 seconds. Once these wall samples were generated, the process of generating the mesh took a total of 3.5 seconds. This step includes data i/o, the floor plan generation, and the 3D extrusion of the floor plan. While our surface carving routine is efficient when compared to other similar techniques, generating a model using 2D information is orders of magnitude quicker. Since the floor plan generation technique can be applied in a streaming fashion to input grid-map data, it could be able to run in real-time for compatible SLAM systems.

The texture-mapping process requires more computation time than the surface reconstruction schemes. Using a single-threaded implementation on the same hardware, texturing the output surface of the voxel carving method as shown in Fig. 12 took 10 hours and 44 minutes. The texture-mapping of the surface generated from the floor plan extrusion approach took 113 minutes. The shorter time to texture-map this surface is due to the far fewer number of elements in the output mesh.

In our earlier work [33], we compared the results shown in Sec. III with the method presented in [5], which performs Marching Cubes on a voxel grid. Here, we compare our reconstruction scheme from Sec. III to static scanning systems, even though they were originally designed to work with ambulatory acquisition systems [10]. As shown in Fig. 11, point-clouds generated from traditional static-scanning technologies can be used as inputs for our reconstruction techniques. The represented scans in Fig. 11a are taken from the VmmlLab dataset of [14]. The original paper for this dataset details segmentation of point-clouds, and not surface reconstruction, but the same group has also developed surface reconstruction approaches [15], [78]. This dataset contains 133 million points from three scan locations, representing a 20 foot $\times$ 30 foot room. We can convert these data to be used by our techniques by treating each scan location as a pose in the path of an assumed mobile system. A comparison to a state-of-the-art method is shown in Fig. 11b, which generates a model of floors, walls, and ceilings also using floor plan generation techniques [15]. This model is represented with 12 triangles. The models shown in Figs. 11c and 11d are constructed from the methods described in Sections III and IV, respectively. The detailed model in Fig. 11c is represented by 6.6 million triangles, while the simple model shown in Fig. 11d is represented by 124 triangles. The remainder of the examples shown in this section are generated from our ambulatory system. Note that the main difference between these two sources of scans is the level of mis-registration noise. Unlike ambulatory systems, which can result in mis-registration up to 27 cm [12], static scanning systems can be accurate to 0.25 cm [15].

While using an extruded floor plan mesh to generate a texture-mapped model is far more computationally efficient, this method has its own set of limitations. By effectively removing the geometry for furniture and other objects in the environment, this method creates a disparity between what is seen in the camera imagery and the reconstructed mesh. As such, the texture for these objects is incorrectly projected onto the wall surfaces behind the objects, as shown in Fig. 13. The advantage of a fully-3D meshing method is the preservation of fine detail in the geometry, while the advantage of the floor plan extrusion method is speed, simplicity, and a higher

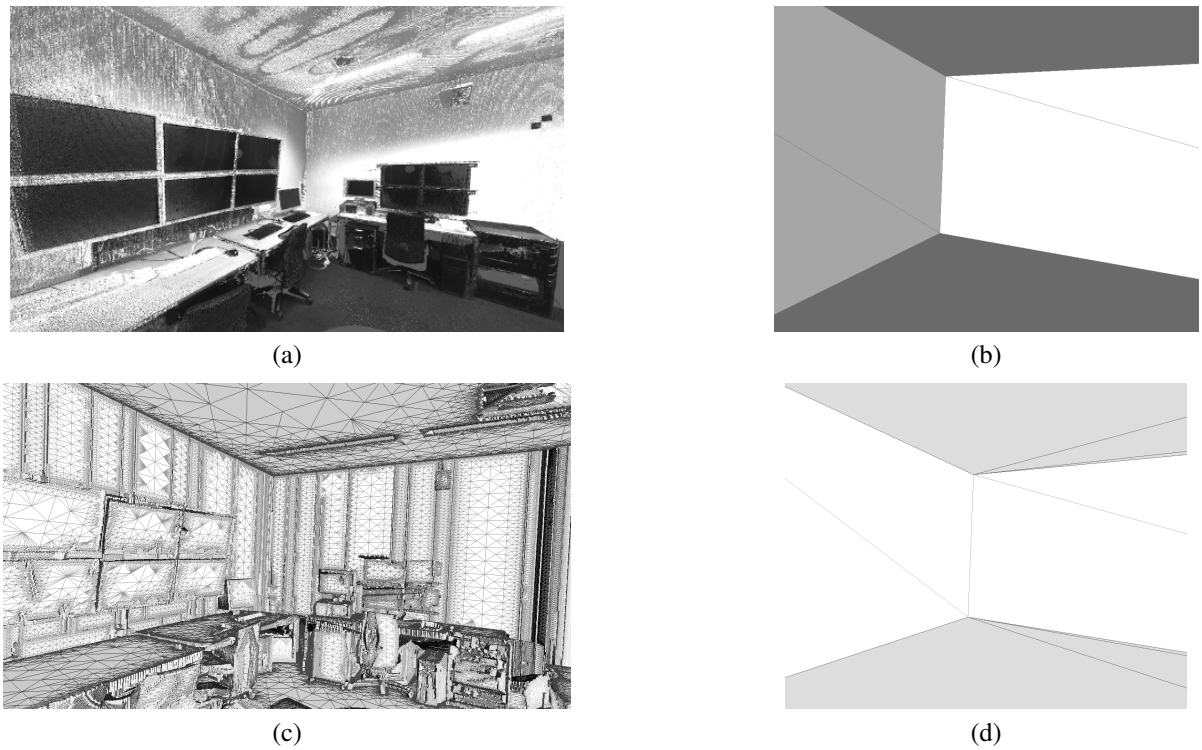


Fig. 11. Comparison to state-of-the-art method: (a) Static point-cloud scans from VmmlLab set of [14]; (b) reconstruction of dataset using method described in [15]; (c) reconstruction of dataset using method from Sec. III; (d) reconstruction of dataset using method from Sec. IV.

proportion of large, planar surfaces that allow for efficient texturing.

VII. Conclusion

Ambulatory systems for mapping building interiors provide advantages in speed and flexibility. They can scan a building area in minutes what would take days with traditional static scanners. The challenge presented is to generate models with the increased noise produced by the localization process for these systems. The modeling approaches presented in this paper are designed to mitigate this noise and to produce results with computational efficiency. Errors in the scans of walls can be reduced by generating floor plans before performing 3D modeling. Uncertainty in the positions of cameras can be counter-acted by refining image locations during texture-mapping. The resulting models are suitable for visualization, simulation, and navigation applications.

References

[1] F. Bosche, "Automated recognition of 3d cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction," *Advanced Engineering Informatics*, vol. 24, pp. 107–118, 2010.

[2] X. Xiong, A. Adan, B. Akinci, and D. Huber, "Automatic creation of semantically rich 3d building models from laser scanner data," *Automation in Construction*, vol. 31, pp. 325–337, 2013.

[3] J. Z. Liang, N. Corso, E. Turner, and A. Zakhor, "Reduced-complexity data acquisition system for image based localization in indoor environments," *IPIN*, October 2013.

[4] Y. Bok, Y. Jeong, and D.-G. Choi, "Capturing village-level heritages with a hand-held camera-laser fusion sensor," *Int J Comput Vis*, no. 94, pp. 36–53, October 2010.

[5] 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.

[6] B. B. Crawley, L. K. Kawrie, C. O. Pedersen, and F. C. Winkelmann, "Energyplus: Energy simulation program," *ASHRAE*, vol. 42, no. 4, pp. 49–56, April 2000.

[7] M. Smith, I. Posner, and P. Newman, "Adaptive compression for 3d laser data," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 914–935, June 2011.

[8] M. F. Fallon, H. Johannsson, J. Brookshire, S. Teller, and J. J. Leonard, "Sensor fusion for flexible human-portable building-scale mapping," *Intelligence Robots and Systems*, pp. 4405–4412, October 2012.

[9] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor, "Indoor localization and visualization using a human-operated backpack system," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*. IEEE, 2010, pp. 1–10.

[10] 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.

[11] 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.

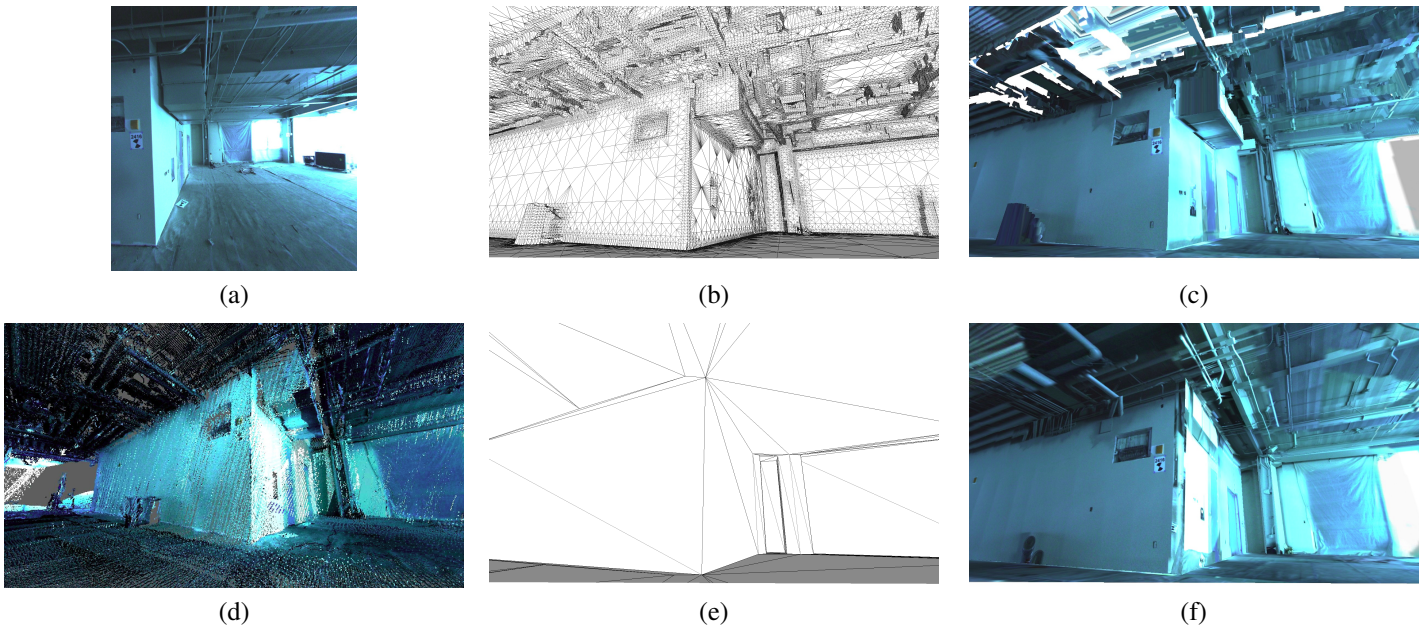
[12] N. Corso and A. Zakhor, "Indoor localization algorithms for an ambulatory human operated 3d mobile mapping system," *Remote Sensing*, vol. 5, no. 12, pp. 6611–6646, October 2013.

[13] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, February 2007.

[14] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornug, and R. Pajarola, "Object detection and classification from large-scale cluttered indoor scans," *Computer Graphics Forum*, 2014.

[15] C. Mura, O. Mattausch, A. J. Villanueva, E. Gobbetti, and R. Pajarola, "Robust reconstruction of interior building structures with multiple rooms under clutter and occlusions," *Proceedings IEEE Conference on Computer-Aided Design and Computer Graphics*, pp. 52–59, 2013.

[16] S. Ochmann, R. Vock, R. Wessel, M. Tamke, and R. Klein, "Automatic generation of structural building descriptions from 3d point cloud scans," *International Conference on Computer Graphics Theory and Applications*, no. 9, January 2014.



(a) (b) (c)

(d) (e) (f)

Fig. 12. Close-up of models generated with the techniques described in this paper: (a) photograph of scanned area; (b) surface carving model from Sec. III; (c) surface carving with textures from Sec. V; (d) point-cloud of scanned area; (e) extruded floor plan model from Sec. IV; (f) extruded floor plan with texturing.

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

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

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[20] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," *IEEE International Conference on Robotics and Automation*, pp. 20–25, 2011.

[21] A. Bachrach, S. Prentice, R. He, P. Hentry, A. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments," *Int. J. Robot. Res.*, no. 31, pp. 1320–1343, 2012.

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

[23] J. A. Baerentzen, "Octree-based volume sculpting," *IEEE Visualization*, 1998.

[24] H. Hoppe, "Progressive meshes," *Computers and Graphics*, 1998.

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

[26] E. Turner and A. Zakhor, "Watertight as-built architectural floor plans generated from laser range data," *3DimPVT*, October 2012.

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

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

[29] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, "Acquiring 3d indoor environments with variability and repetition," *ACM Transactions on Graphics*, vol. 31, no. 6, November 2012.

[30] 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*, vol. 31, no. 137, November 2012.

[31] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an rgbd camera," *ACM Transactions on Graphics*, vol. 31, no. 6, November 2012.

[32] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," *EECV 2012 Lectures in Computer Science*, vol. 7572, pp. 668–681, 2012.

[33] E. Turner and A. Zakhor, "Watertight planar surface meshing of indoor point-clouds with voxel carving," *3DV*, June 2013.

[34] M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially extended kinectfusion," *CSAIL Technical Reports*, July 2012.

[35] Q. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics*, vol. 32, no. 112, July 2013.

[36] ——, "Color map optimization for 3d reconstruction with consumer depth cameras," *ACM Transactions on Graphics*, vol. 33, 2014.

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

[38] K. Denker, B. Lehner, and G. Umlauf, "Real-time triangulation of point streams," *Engineering with Computers*, vol. 27, pp. 67–80, 2011.

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

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

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

[42] ——, "Parallel poisson surface reconstruction," *ISVC*, pp. 678–689, 2009.

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

[44] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, July 1987.

[45] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," *SIGGRAPH*, pp. 209–216, 1997.

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

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

[48] 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*, vol. 27, no. 110, December 2008.

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

[50] Y.-K. Yang, J. Lee, S.-K. Kim, and C.-H. Kim, "Adaptive space carving with texture mapping," *ICCSA*, vol. 3482, pp. 1129–1138, 2005.

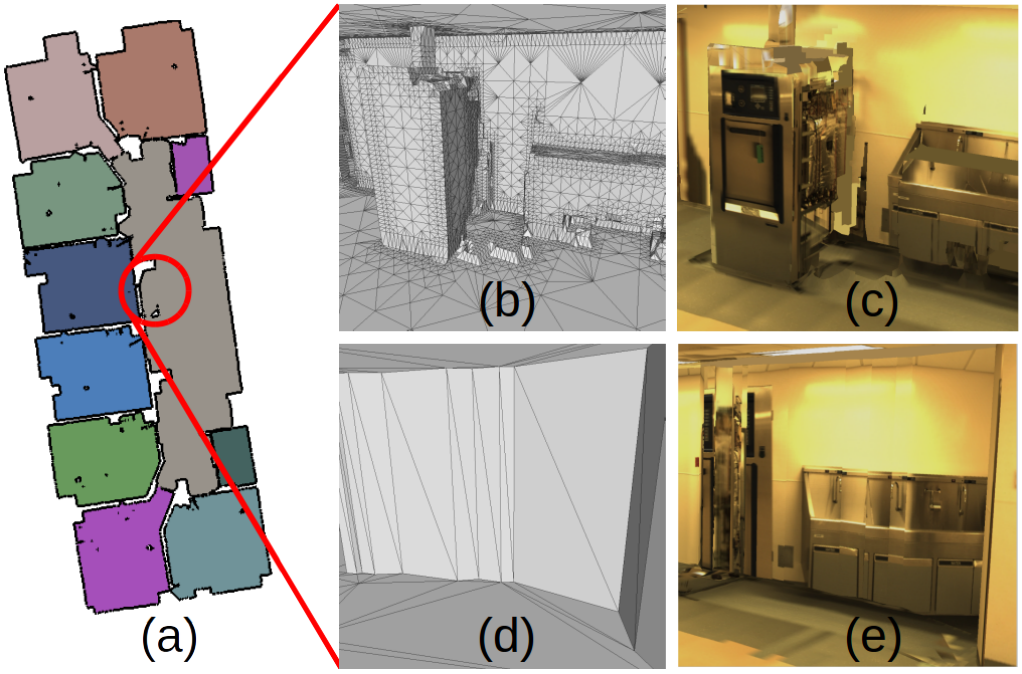


Fig. 13. Example of limitations of texture-mapping: (a) Floor plan of environment generated using method from Sec. IV; (b) surface carving model from Sec. III; (c) surface carving with textures from Sec. V; (d) extruded floor plan model from Sec. IV; (e) extruded floor plan with textures. Note that the objects in the environment are not represented in the mesh of the extruded floor plan, so their texture is projected to the back wall of the room.

[51] K. Zhou, M. Gong, and B. Guo, "Data-parallel octrees for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 5, pp. 669–681, May 2011.

[52] A. Adan and D. Huber, "3d reconstruction of interior wall surfaces under occlusion and clutter," *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pp. 275–281, 2011.

[53] F. Lafarge and P. Alliez, "Surface reconstruction through point set structuring," *Eurographics*, vol. 32, no. 2, 2013.

[54] S.-H. Or, K.-H. Wong, Y. kin Yu, and M. M. yuan Chang, "Highly automatic approach to architectural floorplan image understanding and model generation," *Pattern Recognition*, November 2005.

[55] R. Lewis and C. Sequin, "Generation of 3d building models from 2d architectural plans," *Computer-Aided Design*, vol. 30, no. 10, pp. 765–779, 1998.

[56] B. Okorn, X. Xiong, B. Akinci, and D. Huber, "Toward automated modeling of floor plans," *3DPVT*, 2009.

[57] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," *Proceedings of IEEE International Conference of Robotics and Automation*, pp. 2443–2448, April 2005.

[58] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[59] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," in *ACM Transactions on Graphics (TOG)*, vol. 25. ACM, 2006, pp. 853–861.

[60] L. Wang, S. Kang, R. Szeliski, and H. Shum, "Optimal texture map reconstruction from multiple views," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–347.

[61] S. Coorg and S. Teller, "Matching and pose refinement with camera pose estimates," in *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[62] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 11–20.

[63] F. Bernardini, I. Martin, and H. Rushmeier, "High-quality texture reconstruction from multiple scans," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 7, no. 4, pp. 318–332, 2001.

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

[65] E. Turner and A. Zakhor, "Floor plan generation and room labeling of indoor environments from laser range data," *International Conference on Computer Graphics Theory and Applications*, no. 9, January 2014.

[66] T. A. Funkhouser, C. H. Sequin, and S. J. Teller, "Management of large amounts of data in interactive building walkthroughs," *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pp. 11–21, 1992.

[67] E. Brunskill, T. Kollar, and N. Roy, "Topological mapping using spectral clustering and classification," *International Conference on Intelligent Robots and Systems*, pp. 2491–3496, October 2007.

[68] *Americans with Disabilities Act*, U.S. Architectural and Transportation Barriers Compliance Board, 1331 F Street N.W. Suite 1000 Washington D.C. 20004-1111, July 1990, aNSI A117.1-1980.

[69] P. Cheng, M. Anderson, S. He, and A. Zakhor, "Texture mapping 3d planar models of indoor environments with noisy camera poses," *SPIE Electronic Imaging Conference, Computational Imaging XII*, no. 9020, February 2014.

[70] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, no. 1, pp. 59–73, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-0002-3

[71] M. Brown and D. Lowe, "Autostitch," http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html.

[72] A. S. Glassner, *An Introduction to Ray Tracing*. Academic Press, 1989.

[73] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[74] A. Elqursh and A. Elgammal, "Line-based relative pose estimation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, june 2011, pp. 3049 –3056.

[75] J. Koeck and W. Zhang, "Extraction, matching and pose recovery based on dominant rectangular structures," 2005.

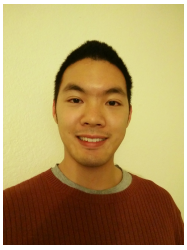
[76] C. Wu, "Siftgpu," http://www.cs.unc.edu/~ccwu/siftgpu/.

[77] E. Turner, "Video of generated models," http://www-video.eecs.berkeley.edu/research/indoor/video/jstsp2014.wmv.

[78] C. Mura, O. Mattausch, A. J. Villanueva, E. Gobbetti, and R. Pajarola, "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts," *Computers and Graphics*, vol. 44, pp. 20–32, November 2014.

Eric Turner (S'14) is a Ph.D. Candidate at UC Berkeley's Video and Image Processing Lab. He received his B.S. degree in Electrical and Computer Engineering from Carnegie Mellon University in 2011, and his M.S. in Electrical Engineering and Computer Sciences from UC Berkeley in 2013.

Eric is a 2013 awardee of the National Defense Science and Engineering Graduate Fellowship. He was awarded best student paper at the 9th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. His research focuses on surface reconstruction of indoor building environments using mobile scanning systems.

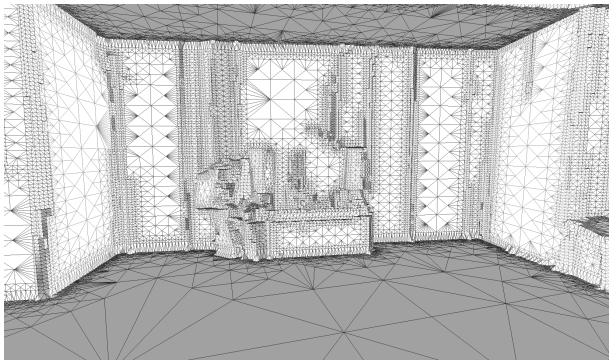


Peter Cheng received his Bachelor's and Master's degrees in Computer Science from UC Berkeley in 2012 and 2013, respectively. While there, he worked on texture generation as part of the Video and Image Processing lab. He is currently an engineer at Amazon Lab126 working on computer vision applications.

Avideh Zakhor (M'87 SM'01 F'02) joined the faculty at UC Berkeley in 1988 where she is currently a professor and holds Qualcomm chair in Electrical Engineering and Computer Sciences. Her areas of interest include theories and applications of signal, image and video processing, 3D computer vision, and multimedia networking. She has won a number of best paper awards, including the IEEE Signal Processing Society in 1997 and 2009, IEEE Circuits and Systems Society in 1997 and 1999, international conference on image processing in 1999, Packet Video Workshop in 2002, and IEEE Workshop on Multimodal Sentient Computing in 2007. She holds 6 U.S. patents, and is the co-author of three books with her students.

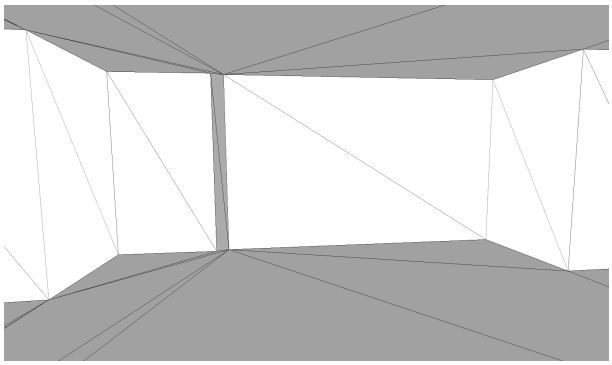
Prof. Zakhor received her B.S. degree from California Institute of Technology, Pasadena, and her S.M. and Ph. D. degrees from Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987 respectively. She was a General Motors scholar from 1982 to 1983, was a Hertz fellow from 1984 to 1988, received the Presidential Young Investigators (PYI) award, and Office of Naval Research (ONR) young investigator award in 1992. In 2001, she was elected as IEEE fellow and received the Okawa Prize in 2004.

She co-founded OPC technology in 1996, which was later by Mentor Graphics (Nasdaq: MENT) in 1998, Truvideo in 2000, and UrbanScan Inc. in 2005 which was acquired by Google in 2007.

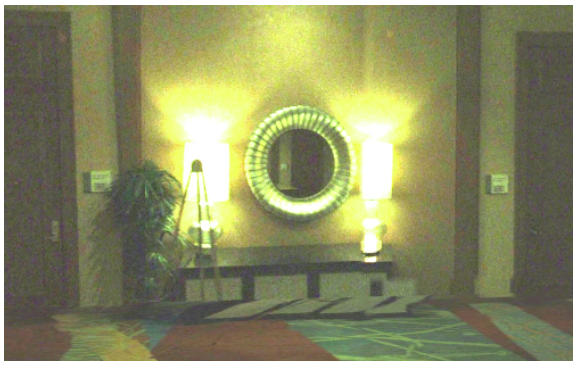


(a)

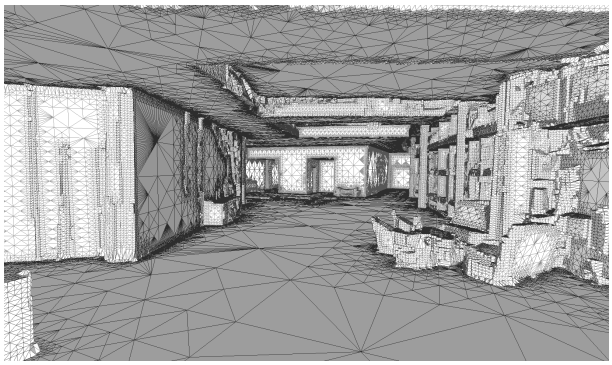
(b)



(c)

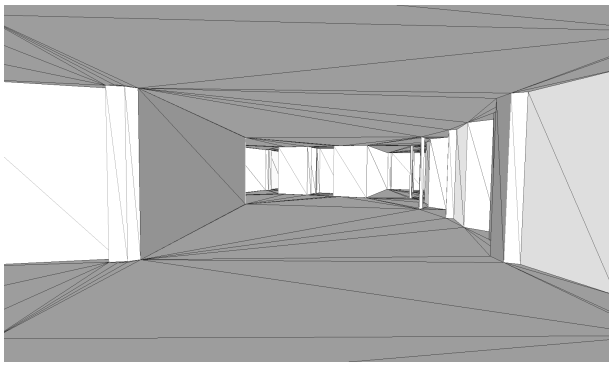


(d)



(e)

(f)



(g)

(h)

Fig. 14. Modeling results of hotel lobby: (a) Desk area modeled with method from Sec. III; (b) texture-mapping from Sec. V applied; (c) area modeled with method from Sec. IV; (d) texture-mapping from Sec. V applied; (e) main lobby area modeled with method from Sec. III; (f) texture-mapping applied; (g) area modeled with method from Sec. IV; (h) texture-mapping applied.