Watertight Planar Surface Reconstruction of Voxel Data

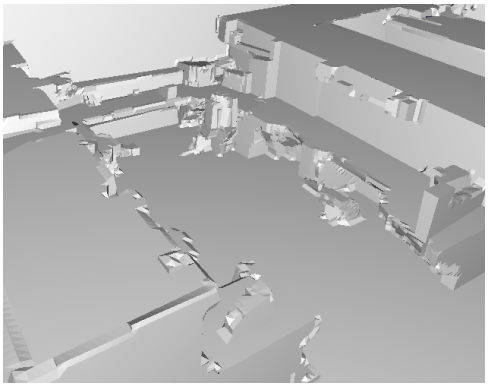
Eric Turner
CS 284
Final Project Report
December 13, 2012

1. Introduction

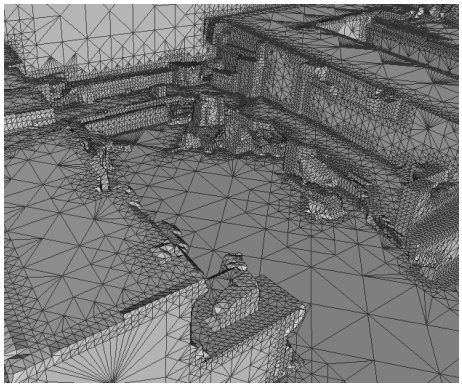
There are many scenarios where a 3D shape is represented by a voxel occupancy grid. Oftentimes it is desirable to convert data from this format to a triangulated mesh that represents the surface of the volume described by the occupied voxels. Algorithms such as Marching Cubes [8] or Dual Contouring [6] can be applied, or each square voxel face could be converted to two triangles directly [9]. Unfortunately, each of these approaches has their drawbacks.

The input voxel grid describes a discretized surface. If this surface represents a flat region that is not axis-aligned, the discretization will result in a zig-zag pattern along the boundary voxels. Such artifacts remain in the output of the above traditional approaches, even though it is typically undesirable. Additionally, these approaches always output elements of uniform size. The output mesh uses many triangles to model flat regions, even though such a region could be modeled just as accurately with a few large elements.

These downsides dramatically affect models that are piece-wise planar. This project proposes an alternate approach. Given a voxelized representation of a volume, the algorithm described below will fit a piece-wise planar surface to these voxels, then form a mesh on each planar regions with proportionally-sized triangles. The result is a watertight reconstruction which accurately represents the underlying shape with sharp features and far fewer elements.



(a)



(b)

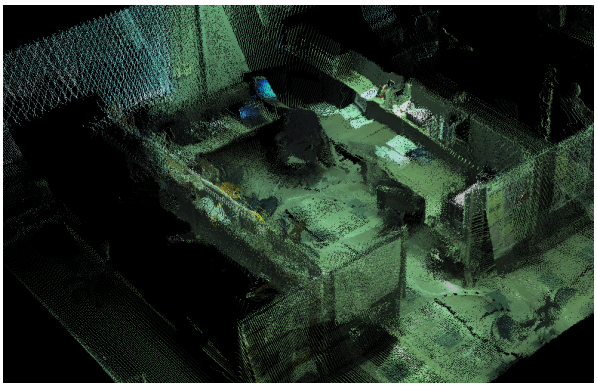
Figure 1. Example output of proposed algorithm (a), with triangulation shown (b). Surface reconstruction from input point-cloud shown in Figure 2.

2. Background

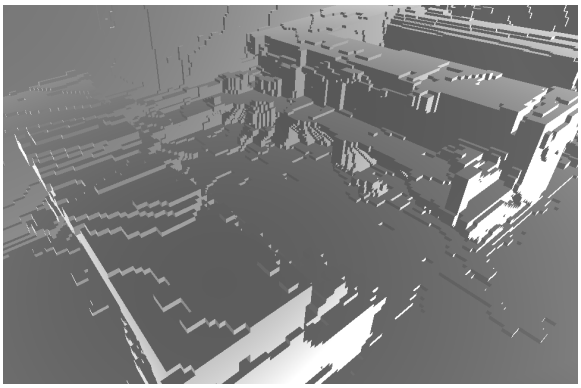
This project is intended for voxelized representations of piece-wise planar shapes. Man-made objects are good examples. This report considers 3D models of the interior architecture of buildings. These models can be recovered with a mobile scanning device, which travels through the hallways and rooms of a building [1]. The scanning system performs Simultaneous Localization and Mapping (SLAM) to recover the path traveled.

At any given instant in time, the system knows its orientation in space. In the same orientation, laser range finders on the scanning system can measure the distance of building architecture from the system. The result is a point-cloud representing all the laser measurements taken during the data collection process. An example point-cloud is shown in Figure 2a.

This point-cloud can be used to create a model of the building. While the point-cloud is a sampling of the building geometry, it is more useful to construct a meshed surface, which can be used for texturing or geometric analysis.



(a)



(b)

Figure 2. Example point-cloud of office building interior (a) and corresponding input voxel boundary (b).

To ensure watertightness of this surface, a volumetric approach is used. The points in the point-cloud are used to determine which locations in the volume are "interior" and which are "exterior". From this partitioning, the dividing surface of these voxels can be exported, as shown in Figure 2b.

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. The laser scans represent line segments originating from the scanner's position at a given point in time, and terminating at the scanned points. If a voxel intersects any of these lines, it is considered interior. Otherwise, it is considered exterior. The boundary between the two types of voxels represents a watertight, closed surface [3]. An example of this process is shown in Figure 2b.

3. Approach

The procedure for surface reconstruction of voxels can be broken into two sections. First, estimates of planar regions are found around the boundary faces of these voxels. These regions are formed by connected sets of voxel faces, all of which are positioned near a best-fit plane. Second,

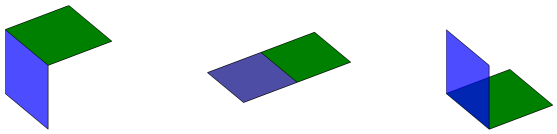


Figure 3. A voxel face (in green) must have a neighbor on its left edge (in blue), in one of three possible positions.

these regions are found, the set of voxel faces is replaced by a triangulation of the same shape of the region on the mesh. This triangulation lies along the best-fit plane, with elements whose size are proportional to the size of the region.

3.1. Region Growing on Voxels

The first task is to determine the connectivity along the input voxel faces. Since these faces form a watertight surface and lie on an axis-aligned grid, all faces will have exactly four neighboring faces, each of which can be in three possible positions, as demonstrated in Figure 3.

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 will lie exactly on a plane.

Since the voxels are a discretized representation of the volume, then it is beneficial to attempt to fit planes that are only approximated by the voxel faces. For any subset of voxel faces, we can compute a best-fit plane to these faces using Principle Components Analysis on the corners of each face [5]. If the maximum error of any of these faces from this plane is at most r, then the discretization of this plane would exactly be this subset of faces.

Using the set of regions found from the flood-fill operation, two regions can be merged if the best-fit plane of this union satisfies the above criterion for the maximum error, as shown in Figure 4. This process is repeated as long as there exist regions that can be merged. The result of this step is a new set of planar regions, whose discretization exactly matches the original voxel faces. This stage typically results in over-fitting of planes, since the original voxel carving may have introduced some error. In order to yield a

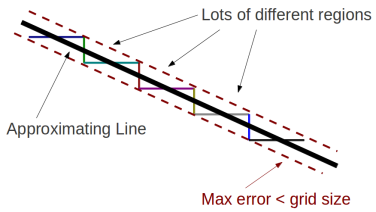


Figure 4. The flood-filled regions are merged as long as the best-fit plane of the merger has a maximum error of r

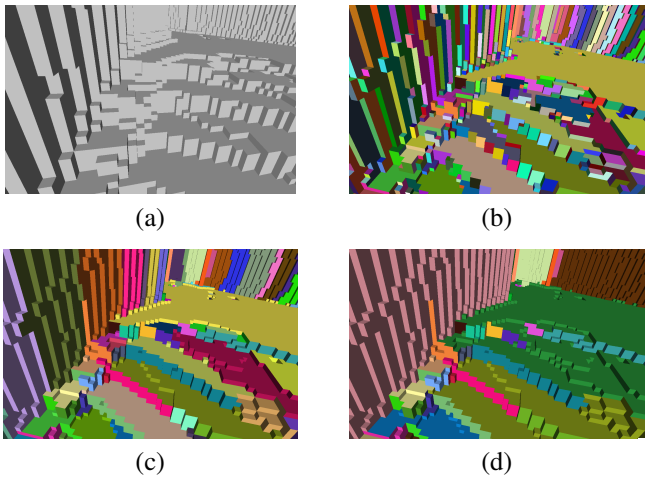


Figure 5. (a) Example carved voxels at the top of a flight of stairs; (b) Regions colored based on voxel face flood-fill; (c) Region growing by finding best-fit planes to voxel faces; (d) Regions relaxed to merge planes that are nearly parallel

more aestetically pleasing output, these region definitions can be further relaxed. This step is performed by locating all adjacent regions whose planes are nearly parallel. If two regions are fit by define planes whose normal vectors are within $15°$, then these two planar regions 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. The steps for region growing are shown in Figure 5.

3.2. Triangulation of Regions

Once the set of voxel faces has been partitioned into planar regions, it is necessary to triangulate these regions. As mentioned above, a number of surface reconstruction algorithms exist for triangulating voxel data [8, 6]. Since the output mesh will represent the planar regions found in the previous section, an optimum approach would adjust the size of elements based on the size of these planar regions.

One option would be to compute a uniform triangulation using a traditional approach, then to simplify this triangulation within each region. This would reduce the number of elements in larger regions. The downside to this approach deals with the limitations of simplification schemes [2, 4]. These simplification schemes often become degenerate when curvature nears zero, such as in a planar region. Additionally, there are no guarantees are made about the quality of the resulting triangles. Further, this approach would require the creation of the many more triangles than would ultimately be used. A more efficient approach would be to adaptively triangulate the planar regions directly and not use an intermediary triangulation.

Taking advantage of the existing voxel grid can help 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 bounds on resulting triangle angles [7]. By performing this triangulation after projecting onto the dominant axis of each region's normal vector, this approach can be used even if the region itself is not axis-aligned.

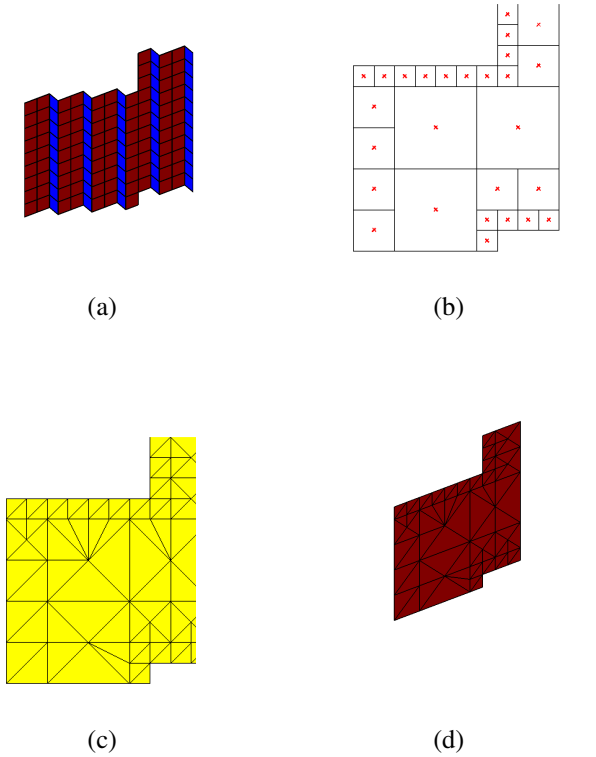


Figure 6. (a) A planar region is composed of voxel faces, with the faces along the dominant axis are shown in red; (b) The dominant faces can be represented in a quadtree structure to reduce number of elements; (c) This quadtree can be triangulated efficiently while ensuring high-quality triangles; (d) The resulting triangulation is projected back to the plane in 3D.

To ensure connectivity of the final triangulation, the voxel faces within each region are divided into two groups. Any face that has a neighboring face in another region is considered a *boundary* face, while the remaining faces are considered *interior* to that region. Once this projection is performed, the triangulation is found by populating a quadtree aligned to the projected grid with the interior voxel faces. The leaves of this quadtree are simplified to represent the same area. Each leaf is triangulated by placing an additional voxel at the center. The result gives larger triangles with larger leaf nodes, while still controlling the quality of the output triangles. This process is shown in Figure 6.

The boundary voxel faces are triangulated simply with two triangles per face. The vertices of these boundary vox-

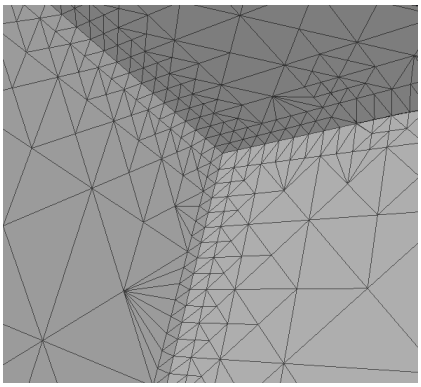


Figure 7. The triangulation of the boundary between three regions in the corner of a room.

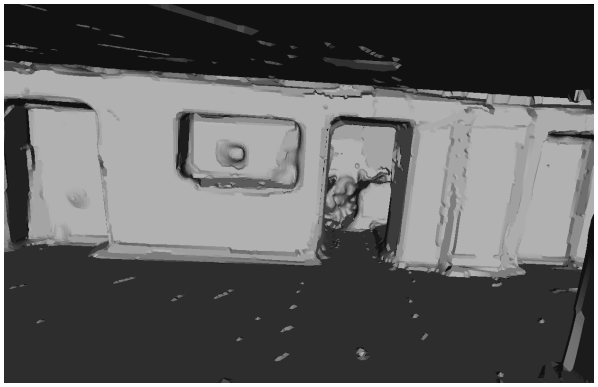
els are then projected onto the best-fit planes of the regions. For vertices that are shared by multiple regions, their position is snapped onto the intersection of those regions. To limit self-intersections in the final surface, the displacement allowed by this snap is limited by a distance threshold. This threshold is relaxed as the angles between the regions in question approach 90°. In this sense, the corners between walls and ceilings are perfectly sharp, while the transition between regions that are close to parallel is smooth. An example of regions intersecting at near right-angles is shown in Figure 7.

4. Results

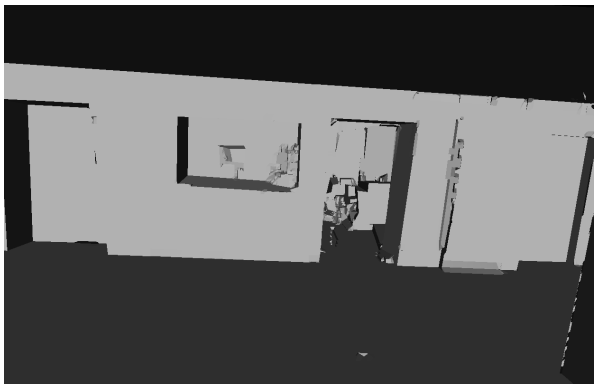
The results of this surface reconstruction routine can be analyzed both qualitatively and quantitatively. In qualitative asseessment, it is most important to verify that walls, floors, and ceilings are immediately identifiable, with sharp creases between them. For quantitative analysis, the resulting mesh is compared to the original point-cloud used to generated the carved voxels. As a control, the characteristics of the proposed method are compared to the results of running Marching Cubes on the same voxel grid, then fitting planar regions to the resultant triangulation. Figure 8 shows a visual 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. The point-clouds shown are colored based on optical imagery recovered during the data collections. Note that for all of these models, large flat areas are represented by fewer, larger triangles. This property can be seen in Figure 1, where the floor and walls have large triangles, whereas the desk area has high detail that is modeled with smaller triangles.



(a)

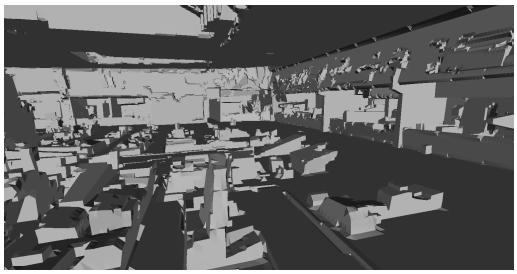


(b)

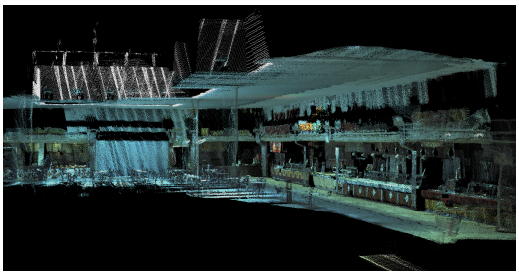
Figure 8. A visual comparison between Marching Cubes (a) and the proposed method (b)

Figure 9 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. The chairs and tables in the center of the room have many details that are well below the 10 centimeter voxel size, so they were not fully captured in the model. A smaller dataset is shown in Figure 10. This model represents a 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. Lastly, Figure 11 shows a modeling of a construction site. Even though the majority of the scanned area is cluttered, the resulting model accurately represents the bounds of the room shown.

Run-time analysis was performed on the dataset shown in Figure 11. The input to this dataset contains 25 million points. 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. Computation

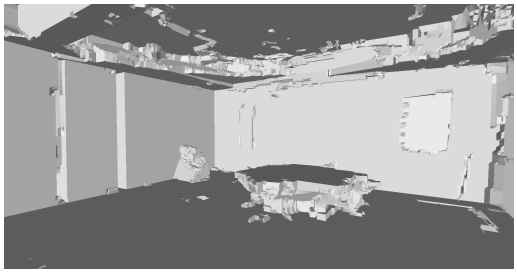


(a)

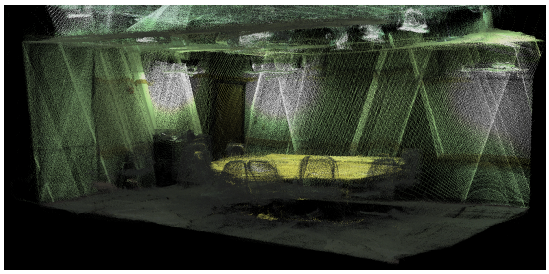


(b)

Figure 9. Surface reconstruction of shopping mall (a), with input point-cloud (b). Resolution 10 cm.



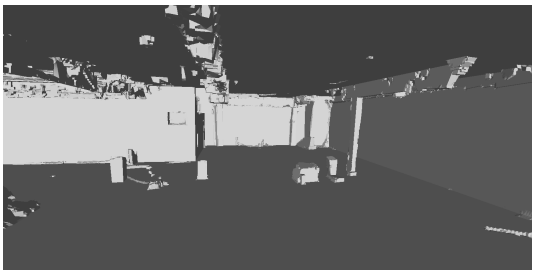
(a)



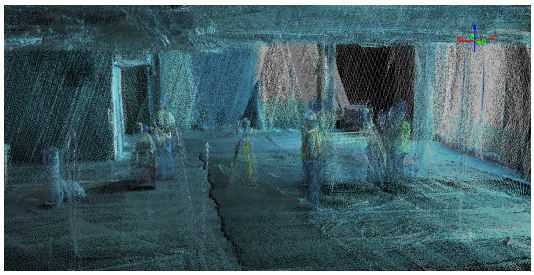
(b)

Figure 10. Surface reconstruction of conference room with table (a), with corresponding point-cloud (b). Resolution 5 cm.

time was also performed for this same dataset with a voxel resolution of 2 centimeters. Run-time of the voxel carving is proportional to volume, and voxel carving took 730 minutes for this resolution, while surface reconstruction took 9.5 minutes.



(a)



(b)

Figure 11. Surface reconstruction of construction site (a) and corresponding point-cloud (b). Resolution 5 cm.

4.2. Mesh Error Analysis

Accuracy of the output mesh is computed with respect to the input point-cloud. The distance of each point to the 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 in the final mesh, since they are the size of one voxel or smaller. These features get carved away on the voxel grid. As a result, even a perfect surface reconstruction of the voxels will have some fixed error with respect to the point-cloud. By fitting regions directly to the voxels, however, instead of triangulating with Marching Cubes and then reducing accuracy with plane fitting to the triangulated mesh, the error of the output surface can be mitigated. As shown in Figure 12, the RMS error of the proposed method is less than that of a traditional marching cubes approach for every resolution. The extent in error savings is more dramatically shown at coarser resolutions.

Since the mesh is an orientable surface, the signed bias of the point-clouds can also be computed. A positive bias indicates that the input point is in front of a triangle inside the mesh, while a negative value indicates that the point is outside the mesh. Since the carving process carves voxels that contain the input point-cloud, a positive bias is expected. As shown in Figure 13, the bias of the proposed method is less than that of Marching Cubes for all voxel resolutions.

5. Conclusions and Future Work

This project is meant to provide a mechanism for converting a point-cloud into a watertight triangulated mesh,

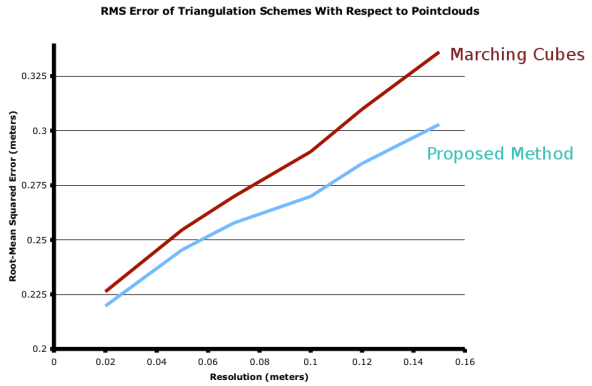


Figure 12. The root-mean-squared error of a triangulation with respect to input point-cloud.

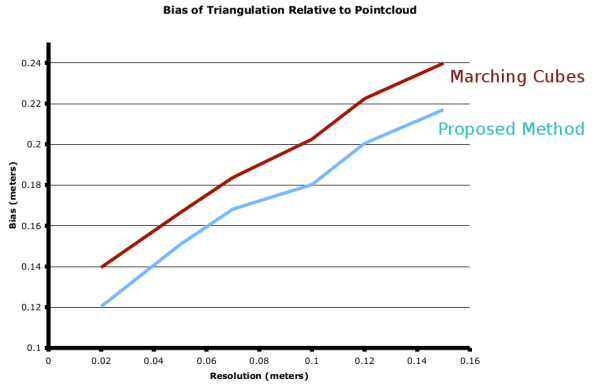


Figure 13. The bias of a triangulation with respect to input point-cloud.

with special consideration to modeling planar regions. Most surface reconstruction techniques focus on exporting smooth and organic-looking models, so this approach can benefit models of man-made objects that are often composed of flat regions and sharp corners. Using a voxelized volumetric representation as the input allows for a watertight output. By performing planar fitting on the input voxel faces before triangulation, the triangles can be adapted to the best-fit planes, resulting in fewer elements. This algorithm is novel in that it converts from voxels to a triangulated mesh with a focus on sharp features, whereas previous work has only focused on organic and smooth features.

The most potent lesson learned from this work has been how much enforcing sharp features leads to difficulty in maintaining mesh quality. Using organically-shaped surfaces typically allow for more easily obtained quality in the mesh, which is typically dependant on local feature size and curvature. When introducing sharp corners and flat regions, local curvature quickly becomes ill-defined. In this project, good quality triangles were acheived by taking advantage of the pre-existing grid structure of the input voxels. While this approach mitigates self-intersections and gaps in the surface, it results in a more complicated approach than the always-favorite of marching cubes.

There are a number of possible future extensions to this project. Currently, planar regions are grown in a single pass. One could use an iterative K-means approach to adjust the extent of the regions until convergence is reached. This approach is not included in this report mainly for computational issues. Currently, finding the extent of the regions can take three or four minutes on the larger datasets, which have millions of triangles and hundreds of thousands of regions. A K-means approach would require many iterations of this step. Additionally, while the number of elements used in the proposed algorithm is fewer than in Marching Cubes, there are locations where more elements than necessary are used, such as the boundary between two large regions (see Figure 7). Once watertightness is assured, these locations could be simplified to reduce total number of elements.

References

[1] 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. 1

[2] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *SIGGRAPH*, pages 209–216, 1997. 3

[3] 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. 2

[4] H. Hoppe. Progressive meshes. *Computers and Graphics*, 1998. 3

[5] I. T. Jolliffe. *Principal Components Analysis, Second Edition*. Springer, 1986. 2

[6] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *SIGGRAPH*, 2002. 1, 3

[7] F. Labelle and J. R. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, August 2007. 3

[8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4), July 1987. 1, 3

[9] Y.-K. Yang, J. Lee, S.-K. Kim, and C.-H. Kim. Adaptive space carving with texture mapping. *ICCSA*, 3482:1129–1138, 2005. 1