Robust Meshes from Multiple Range Maps

Kari Pulli* Tom Duchamp* Hugues Hoppe†
John McDonald* Linda Shapiro* Werner Stuetzle*
*University of Washington, Seattle, WA
†Microsoft Research, Redmond, WA

Abstract

This paper presents a method for modeling the surface of an object from a sequence of range maps. Our method is based on a volumetric approach that produces a compact surface without boundary. It provides robustness through the use of interval analysis techniques and computational efficiency through hierarchical processing using octrees.

1. Introduction

Surface reconstruction from range data involves four major steps.

Step 1: Data acquisition. Range data sets covering the surface to be modeled are obtained. Some preprocessing of the data may be required, such as low-pass filtering and removal of data points that belong to other objects or background.

Step 2: Registration. In general, each range view is in its own coordinate system. The collection of views (range maps) are registered into a common object-centered coordinate system.

Step 3: Integration. The separate registered range maps are integrated into a single surface representation (often a polygon mesh).

Step 4: Optimization. The single surface representation can be better fitted to the data, it may be further simplified, or the representation can be converted to some other format (e.g., smooth surfaces).

In this paper we propose a novel solution to Step 3. Our method proceeds in two steps: it first hierarchically builds a volumetric representation of an object, and then extracts a triangle mesh from the volumetric representation. Our method is efficient in computation time, is robust against outliers, and automatically fills in small holes in range images due to gaps in the data. It can thus quickly recover the topology of arbitrary surfaces, even in the presence of outliers and missing data.

In Section 2 we describe our algorithm. We present our results in Section 3, and we discuss our method and previous work in Section 4. Section 5 concludes the paper.

2. Algorithm

Our algorithm processes a cubical volume surrounding all the input data in a hierarchical fashion. For each cube-shaped partition, it checks whether the cube can be shown to be entirely inside or outside of the object. If neither, the cube is subdivided and the same test is recursively applied to the resulting smaller cubes. Our surface approximation is the closed boundary between cubes that lie entirely outside of the object and all the other cubes.

2.1. Assumptions

We assume that the range data is expressed as a set of range maps called views. Each view is like a 2D image, except that at each pixel a 3D point is stored instead of a color value. Further, we assume that the calibration parameters of the sensor are known so that we can project any 3D point to the image plane of the sensor. We also assume that the line segment between the sensor and each measured point lies entirely outside of the object we are modeling. Finally, we assume that all range views have been registered to a common coordinate system.

2.2. Processing a single range view

The initial volume is an axis-aligned cube that fully surrounds all the range data. We use interval analysis to evaluate the volumetric function on the cube. If the cube is neither completely inside nor completely outside the object, we recursively subdivide it into eight smaller cubes, which are added to the octree as children of the current cube. For each of these cubes, we classify their location with respect to the sensor and the range data. Note that the initial cube

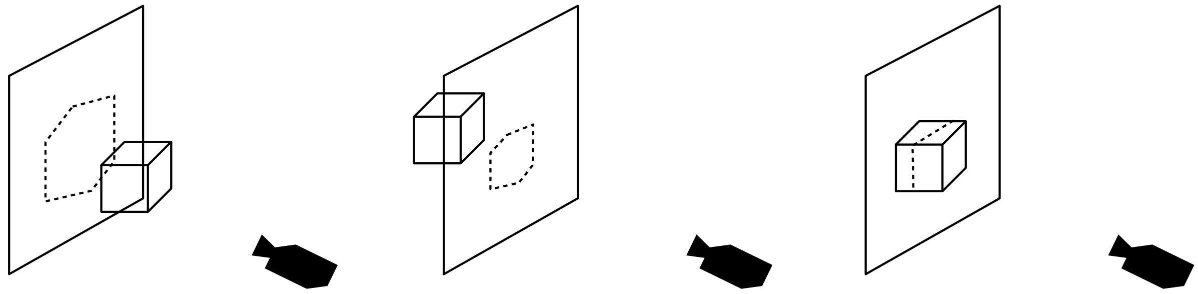


Figure 1. The three cases of the algorithm. In case 1 the cube is in front of the range data, in case 2 it is entirely behind the surface (with respect to the sensor), while in case 3 the cube intersects the range data.

is always subdivided since it is by definition neither inside nor outside the object.

There are three possibilities (see Figure 1).

- In case 1, the cube lies between the range data and sensor. The cube is assumed to lie outside of the object. It is not processed any further.
- In case 2 the whole cube is behind the range data. As far as the sensor is concerned, the cube will be assumed to lie inside of the object. It will not be further subdivided.
- In case 3 the cube intersects the range map. In this case we subdivide the cube into its eight children and recursively apply the algorithm up to a pre-specified maximum subdivision level. A case 3 cube at the finest level is assumed to be at the boundary of the object.

The cubes are labeled as follows. We project the eight corners of the cube to the sensor's image plane, where the convex hull of the points forms a hexagon. The rays from the sensor to the hexagon form a cone, which we truncate so that it just encloses the cube (see Figure 2). If all the data points projecting onto the hexagon are behind the truncated cone (i.e., are farther than the farthest corner of the cube from the sensor), the cube is outside. If all those points are closer than the closest cube corner, the cube is inside. Otherwise, we are in the boundary case. Possible missing data is treated as points that are very close to the sensor.

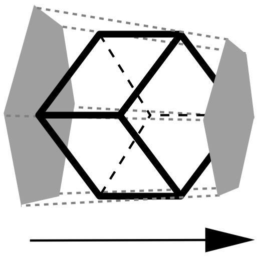


Figure 2: An octree cube and its truncated cone. The arrow points to the sensor.

Our labeling method is simple, but conservative. For instance, if all the points projecting onto the hexagon are actually behind the cube, but some of them are inside the truncated cone, we would erroneously label the cube to intersect the surface. This, however, is not a problem because the next subdivision level will most likely determine that all the children of the cube lie in the exterior of the object and therefore remove them.

We could make our test tighter by performing a more careful test for points that are within the truncated cone: For each such point, determine the two faces of the cube intersected by the ray associated with the point and test whether the faces are behind, in front of, or around the point. It may make sense to perform this extra test at the last subdivision level.

2.3. Generalizations to multiple views

If multiple views are available, we have a choice of two processing orders. We can traverse the whole octree once and use the consensus of all the views to determine the labeling for each cube (simultaneous processing), or we can process one view at a time, building on the results of the previously processed views (sequential processing).

In simultaneous processing, we traverse the octree as we did in the case of a single view. However, the cube labeling process changes slightly.

- A cube is labeled to be inside the object only if it would be labeled inside with respect to each single view.
- A cube is labeled outside if it would be labeled outside with respect to any single view.
- Otherwise, the cube is labeled boundary and is further subdivided, unless the maximum subdivision level has been reached.

We use sequential processing if we later obtain a new view that we want to integrate into a previously processed

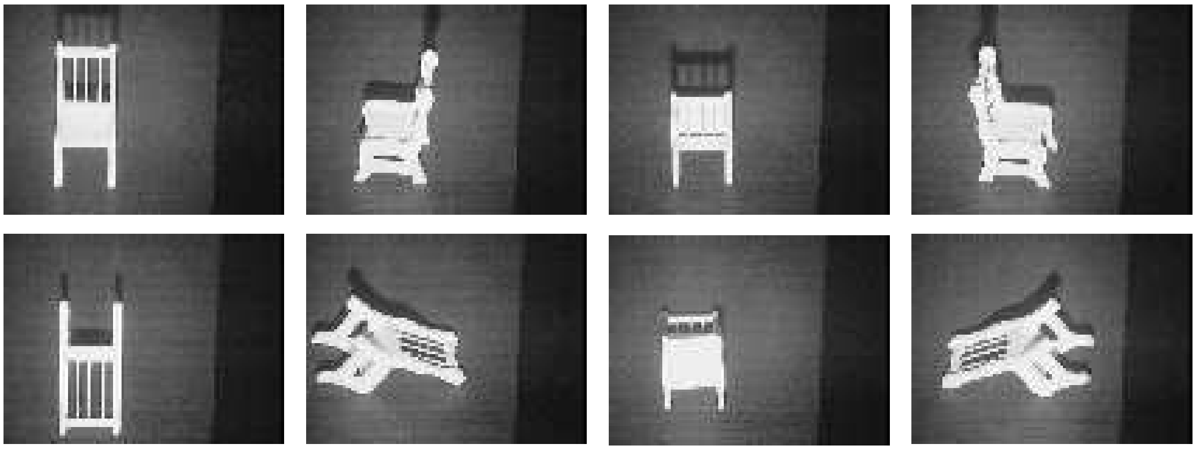


Figure 3. Eight intensity images corresponding to the views of the miniature chair.

octree. We recursively descend the octree and perform the occlusion test for each cube that has not been determined to lie outside of the object. If the new view determines that a cube is outside, it is relabeled and the subtrees below it are removed. Similarly, a boundary label overrides a previous inside label, in which case the cube's descendants have to be recursively tested, potentially up to the maximum subdivision level.

Although both processing orders produce the same result, the simultaneous processing order is in general faster [12]. In sequential processing the silhouette of the object often creates a visual cone (centered at the sensor) that separates volumes known to be outside from those speculated to be inside. We would then have to recurse up to the finest subdivision level to accurately determine this boundary. In simultaneous processing, however, another view could determine at a rather coarse level of subdivision that at least part of that boundary is actually outside of the object, and the finer levels of the octree for that subvolume need never be processed.

Although slower, the sequential processing approach has the advantage of being more memory efficient because it only uses the data from a single view at a time.

In cases where a large number of views of the same object are given, we can use a hybrid approach in which we divide the views into smaller sets and sequentially apply simultaneous processing to each of the smaller data sets.

2.4. Pruning the octree

Due to our conservative label test, or if we use sequential processing, we may determine that all the children of a cube lie in free space. Whenever this happens the eight sibling cubes are recursively collapsed into their parent, which is then labeled also to be outside the object.

2.5. Mesh extraction

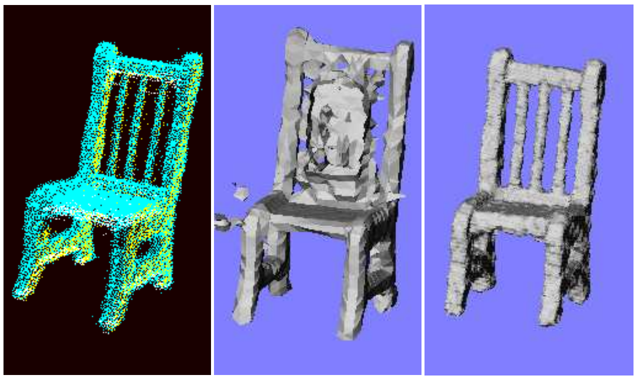
The labeling in the octree divides the space into two sets: the cubes known to lie outside of the object and the cubes that are assumed to be part of the object. Our surface estimate will be the closed boundary between these sets. This definition allows us to create a plausible surface even at locations where we failed to obtain data [4]. The boundary is represented as a collection of vertices and triangles that can be easily combined to a mesh.

The octree generated by the algorithm has the following structure: outside cubes and inside cubes do not have any children, while the boundary cubes have a sequence of descendants down to the finest subdivision level. We traverse the octree starting from the root. At an outside cube we do nothing. At a boundary cube that is not at the finest level, we descend to the children. If we reach the maximum subdivision level and the cube is either at the boundary or inside we check the labeling of the six neighbors. If a neighboring cube is an outside cube, we create two triangles for the face they share. In an inside cube that is not at the maximum subdivision level, we check whether it abuts with an outside cube, and in such case create enough triangles (of same size as the ones created at the finest level) to cover the shared part of the face.

In order to avoid producing multiple copies of the same vertex, the vertices are put into a hash table and a new vertex is created only if it does not already exist. The triangles are combined into a closed triangle mesh.

3. Results

We have tested our method with both real and synthetic data. The real data set consisted of eight views of a miniature chair (Figure 3). Figure 4(a) shows the data points



(a) (b) (c)

Figure 4. (a) The registered point set. (b) The result from using the method of [6]. (c) The result from our method.

(registered using a slightly modified version of Chen and Medioni's method [1]) and the failed surface reconstruction that was our original motivation for this method. Even though we have cleaned the data and removed most of the outliers, we still have some noisy measurements close to the surface, especially between the spokes of the back support of the chair. The algorithm from [6] does not use any knowledge (such as viewing directions, etc.) of the data except the points, themselves. It works quite nicely if the data does not contain outliers and uniformly samples the underlying surface. Unfortunately real data, such as this data set, often violate both of those requirements. Figure 4(b) shows the result. In contrast, Figure 4(c) shows the result of our method.

Figure 5 shows intermediate results of our method using the chair data set displaying the octree after 4, 5, 6, and 7 subdivisions. Figure 6 presents some statistics for processing the chair data set. The final mesh in Figure 4(c) is obtained from the level 7 octree. We smooth the mesh before displaying using Taubin's method [13]. Notice that the spokes and the holes between them have been robustly recovered despite the large number of outliers and some missing data. This is partially due to the implicit removal of outliers that the algorithm performs: in most cases, there is a view that determines that the cube containing an outlier lies outside of the object.

Figure 7 shows the results from a synthetic data set. We generated eight simulated range maps of a temple, registered the views, and applied our algorithm to them. The left figure shows the smoothed result, while the right figure shows the result after applying Hoppe's mesh optimization algorithm [7].

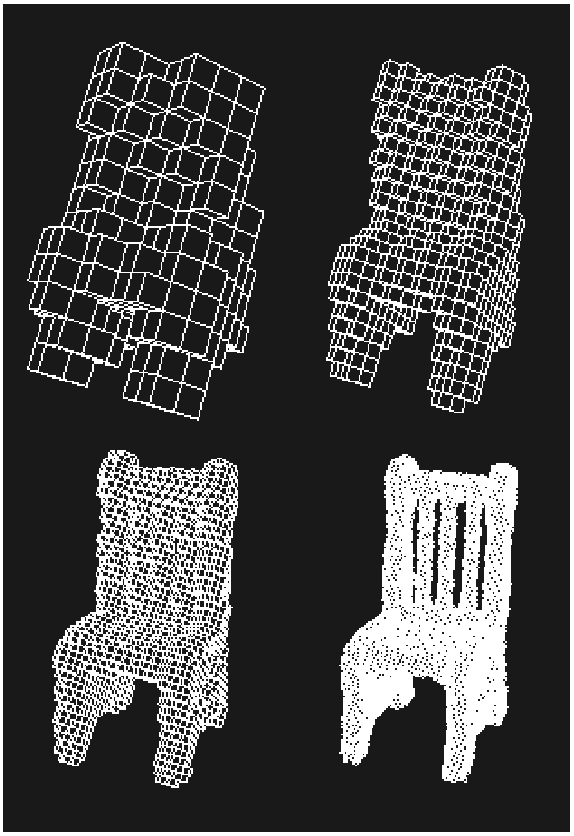


Figure 5. The chair octree after 4, 5, 6, and 7 subdivisions.

levels	nodes	faces	seconds
4	649	422	9.0
5	2377	1372	6.6
6	9393	5152	27.1
7	42657	18446	127.8

Figure 6. Statistics for Figure 5. For example, at 5 levels of subdivision the octree contained 2377 nodes, the boundary between the object and free space was 1372 faces (twice as many triangles), and it took an additional 6.6 seconds to process from the level 4 octree.

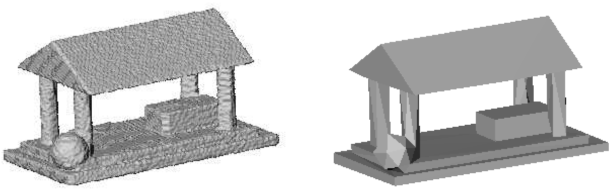


Figure 7. Synthetic temple.

4. Discussion

4.1. Previous work

A popular approach to data integration has been to directly build surfaces from the data. Turk and Levoy [14] and Rutishauser *et al.* [9] create a polygon mesh for each view; the individual meshes are then connected to form a single mesh covering the whole object. Soucy and Laurendeau [11] divide the range data into subsets based on which surface regions are visible within each view. In each subset, the redundancy of the views is used to improve the surface approximation. Finally, the triangulated non-overlapping subsets are connected into a single mesh. Pito [8] integrates meshes by determining where the meshes overlap, choosing the most reliable measurements, and connecting the patches using a modification of [9]. Hoppe *et al.* [6] start from an unorganized set of points, define a signed distance function from the point set, and extract a mesh approximating the zero set of that function. The mesh is iteratively fitted more closely to the data and simplified [7]. The simplified mesh is finally used as a starting point for fitting a piecewise smooth surface representation to the data [5].

Curless and Levoy [4] define a volumetric function based on an average of signed distances to each range image. Their scheme discretely evaluates this volumetric function at points on a uniform 3D grid, and uses these discrete sample values to determine a surface that approximates the zeroset of the volumetric function. As a consequence, their scheme may fail to detect features smaller than the grid spacing and has difficulties with thin features. Also, it requires a significant amount of space, although this can be alleviated through run-length encoding techniques.

Szeliski also used volumetric ideas, constructing octree bounding volumes of the object from silhouettes extracted from intensity images [12]. Octrees have also been used to generate volumes and surfaces from range data. Chien *et al.* [2] combine range quadtrees generated from six orthogonal viewing directions into an octree. Connolly [3] creates an octree representation by organizing the range views into quadtrees and projecting them to the octree, marking the free space along the way.

Of the previous work, Connolly's [3] is closest to ours, but it differs from ours in two important ways. Whereas we project the octree cubes to the depth maps, Connolly does the converse: he first converts the depth maps into quadtrees and then projects them to the octree. Four rays are projected from each vertex of a quadtree node to the octree, and each octree node visited by a ray is removed. Since the octree nodes that are traversed are roughly the size of the quadtree node, the hole that is carved is jaggy and larger than it should be. The carving could be made more accurate by performing the carving at a finer level of resolution, but then the processing becomes more costly, and it becomes more difficult to guarantee that all cubes that should be removed are removed. The organization of the range maps to quadtrees speeds up the processing only outside the visual cone of the object. Unless the object contains many consecutive points that are equidistant from the sensor (an unlikely event), the quadtree becomes fragmented. This causes a great number of holes to be carved into the octree at a fine level of resolution. Both these problems (too much carving and carving at unnecessarily fine resolution) are avoided in our method. The second major difference between the methods is that Connolly's method never removes nodes that have been previously determined to contain a range measurement. We explicitly want to allow this, since that is our mechanism for getting rid of outliers. That is, Connolly gives precedence to surface, while we give precedence to a proof of empty space.

Like Curless and Levoy, we also define a volumetric function (ours is a simple inside/outside binary function), and seek an approximation of the surface that separates the inside and outside regions. But, rather than discretely evaluating the volumetric function on a set of points, our approach is based on an interval analysis technique [10]. That is, we partition space into regions (cubes) and conservatively determine for each region whether it lies completely inside, completely outside, or neither. Using recursive subdivision, we efficiently prune away large regions of space away from the object boundary, and focus both computation and storage right on the spatial region of interest, that is, the region next to the surface.

The interval analysis techniques we use provide improved robustness to our algorithm. To our knowledge, ours is the first approach to use these techniques to range map integration.

4.2. Level of resolution

All volumetric methods require a choice of resolution level. This choice provides a trade-off between the accuracy of the result and requirements for both time and storage. In octree methods such as ours, the resolution is determined by the maximum depth of the octree and it must be specified by the user.

Our algorithm facilitates the optimum choice of resolution appropriate to a given data set. For instance, if we notice that our choice was too coarse to correctly capture the object's topology correctly, we can easily use the results of the previous runs and continue the process with increased choice of maximum depth to achieve a finer resolution.

Of course, the maximum depth is limited by the point density of the data set. Once an octree cube projects to only one or two pixels on the sensor's image plane, the method becomes less robust. A good rule of thumb is that the octree cubes should be larger than each of the following three measures: surface sampling density, sampling error, and registration error.

4.3. Removal of outliers and background

We note earlier that we in effect remove outliers from a view if another view testifies that the outlier lies in free space. This observation shows that the algorithm can be used to automatically remove background and other objects from the views. Suppose that the views were obtained by using a stationary range scanner and by reorienting the object between the views and that we have registered the views into a common coordinate system using a subset of surface points. Now the background and other objects move relative to the object from view to view, and in general some view can determine that they lie outside the object, in which case the points can be labeled as background or removed altogether.

Outliers that lie behind the surface as seen from the scanner can be damaging, since they could cause our algorithm to incorrectly carve away a piece of the object. For several reasons, this has not been a problem for us. Most outliers we have observed either belong to other objects in the background, or appear around the edges of the object. We use background data to carve the space around the object more efficiently, and outliers at the object boundaries do not cause deep holes to be carved into the object.

Obtaining data points from the background is very important for correctly detecting holes. Curless and Levoy [4] advocate the use of backdrops to solve this problem. However, with several types of scanners based on triangulation it is extremely difficult to get measurements through a narrow hole. That was the case in several views of the chair data set. In such cases we can manually paint parts of the missing data to be background.

4.4. Thin objects

Some methods that employ a signed surface distance functions are unable to correctly reconstruct thin objects. Curless and Levoy [4], for example, build a distance function by storing positive distances to voxels in front of the surface and negative distances to voxels behind the surface. In addition to the distance, weights are stored to facilitate combining data from different views. With this method, views from opposite sides of a thin object interfere and may cancel each other, preventing reliable extraction of the zero set of the signed distance function.

Our volumetric method carves away the space between the sensor and the object and does not construct a signed distance function. In case of a thin sheet of paper, our algorithm would construct a thin layer of octree cubes (voxels) straddling the paper. Note however that our method can fail in the presence of measurement noise and registration error if the minimum cube size is set too small. Figure 8 illustrates this phenomenon.

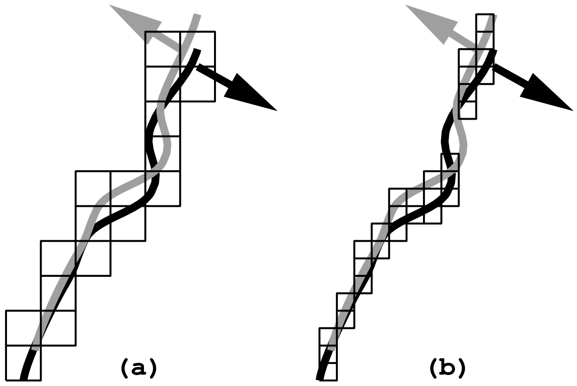


Figure 8. A thin sheet seen from left (gray) and right (black) is reconstructed correctly in (a). In (b) registration error and small cube size combine to cause a hole.

5. Conclusion

We have presented a method for robustly producing a mesh from a set of range views of an object. The method combines the robustness of the volumetric approach [4] with the speed and small memory requirements of octree methods [2, 3, 12]. Our method automatically fills holes due to gaps in input data, it is robust against outliers, it allows incremental addition of range views, and it can model objects of arbitrary topological type, even if the object is thin. A more accurate and/or more concise surface representation can be easily obtained by applying the optimization algorithms described in [7, 5] to the output of our algorithm.

Acknowledgments

We would like to thank Prof. Patrick Flynn for the chair data sets.

This work was supported in part by the National Science Foundation under grants DMS-9492734 and IRI-9520434, grants from Academy of Finland, Finnish Cultural Foundation, and Emil Aaltonen Foundation, as well as industrial gifts from Interval, Microsoft, Xerox, and Human Interface Technology Laboratory.

References

[1] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.

[2] C. H. Chien, Y. B. Sim, and J. K. Aggarwal. Generation of volume/surface octree from range data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '88)*, pages 254–260, June 1988.

[3] C. I. Connolly. Cumulative generation of octree models from range data. In *Proc. Int. Conf. on Robotics and Automation*, pages 25–32, March 1984.

[4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH '96*, pages 303–312, August 1996.

[5] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH '94*, July 1994.

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

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of SIGGRAPH '93*, pages 19–26, August 1993.

[8] R. Pito. Mesh integration based on co-measurements. In *Proc. IEEE Int. Conf. on Image Processing, Special Session on Range Image Analysis*, 1996.

[9] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images of arbitrarily shaped objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 573–580, 1994.

[10] J. Snyder. Interval analysis for computer graphics. In *Proceedings of SIGGRAPH '92*, pages 121–130, July 1992.

[11] M. Soucy and D. Laurendeau. A dynamic integration algorithm to model surfaces from multiple range views. *Machine Vision and Applications*, 8(1):53–62, 1995.

[12] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.

[13] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH '95*, pages 351–358, August 1995.

[14] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94*, pages 311–318, July 1994.