Consensus Surfaces for Modeling 3D Objects from Multiple Range Images

Mark D. Wheeler
Apple Computer, Inc.
1 Infinite Loop, MS: 302-3VR
Cupertino, CA 95014
mdwheel@apple.com

Yoichi Sato
Department of Electrical Engineering and Electronics
Institute of Industrial Science
The University of Tokyo, Tokyo, Japan
ysato@cvl.iis.u-tokyo.ac.jp

Katsushi Ikeuchi*
ki@cvl.iis.u-tokyo.ac.jp

Abstract

In this paper, we present a robust method for creating a triangulated surface mesh from multiple range images. Our method merges a set of range images into a volumetric implicit-surface representation which is converted to a surface mesh using a variant of the marching-cubes algorithm. Unlike previous techniques based on implicit-surface representations, our method estimates the signed distance to the object surface by finding a consensus of locally coherent observations of the surface. We call this method the consensus-surface algorithm. This algorithm effectively eliminates many of the troublesome effects of noise and extraneous surface observations without sacrificing the accuracy of the resulting surface. We utilize octrees to represent volumetric implicit surfaces—effectively reducing the computation and memory requirements of the volumetric representation without sacrificing accuracy of the resulting surface. We present results which demonstrate that our consensus-surface algorithm can construct accurate geometric models from rather noisy input range data.

1 Introduction

In this paper, we present a novel approach for building a 3D surface model from a number of range images of an object. Combining the observations from multiple range image views is an increasingly important problem for automatically modeling 3D objects as range image sensors (e.g., stereo, structured light, laser-based scanners) are becoming widely available. Recent work has made great strides toward this end.

In this paper, we present a method which improves upon previous methods in terms of accuracy and robustness with respect to typical noise characteristics which are common in current range sensing technology. The goal of this work is to use real images of an object to automatically create a model which is:

- Geometrically accurate: depicts the correct dimensions of the object and captures small details of the object geometry
- Clean: eliminates noise and errors which are present in the range image data
- Complete: models the surface as much as is observable from the sample views

We begin by reviewing the three methods which are most closely related to our work. This list is certainly not complete; we refer the reader to [Whe96] for a more complete description of previous work in this area. The following three works and our work make use of volumetric, implicit-surface representation and the marching-cubes algorithm [LC87] to merge the range-image data from several views into a surface model. These algorithms differ in their methods for computing the implicit distance from each voxel (volume element) to the object surface.

Hoppe et al. [HDD+92] were the first to propose constructing 3D surface models by applying the marching-cubes algorithm [LC87] to a discrete, implicit-surface function generated from a set of range images. Their algorithm computes the signed distance function from a set of 3D points. Much of their algorithm works to infer local surface approximations from clouds of points. Local search of the inferred surfaces is used to compute the signed distance from each voxel to the surface of the point set. This method has two problems which limit its effectiveness. First, the algorithm assumes that all of the data belongs to the surface of the object being modeled—large errors will become part of the resulting model. Second, their algorithm oversmooths high curvature regions. This occurs because their algorithm infers surfaces locally from a cloud of points using tangent plane estimation. Thhis method oversmooths high curvature regions as the cloud becomes non-planar in shape.

Curless and Levoy [CL96] followed Hoppe's general scheme with a few significant departures. Rather than performing a search for the closest point from a

*This research has been supported in part by the Advanced Research Projects Agency under the Department of the Army, Army Research Office grant number DAAH04-94-G-0006, and in part by the Office of Naval Research grant number N00014-93-1-1220. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Department of the Army, the Department of the Navy or the United State Government.

voxel's center to determine the signed distance, they integrate distance estimates at each voxel. They take each range image separately and scan along the line of sight to each range image point, integrating the signed distance to that point for each voxel the line passes through. The final signed distance estimate is a weighted average of these estimates. Since the signed distance along the line of sight is integrated on both sides of the observed point, this integration method precludes the modeling of thin surfaces. In addition, their integration method will present a biased estimate of the signed distance even for perfect range data. Unless the line of sight for a particular surface point is perpendicular to the surface, the distance estimate will necessarily be longer than the actual shortest distance to the surface. The error of the distance estimates which contribute to a voxel's estimate may be arbitrarily large. The result of this integration strategy will be inaccuracies in locating the implicit surface.

Hilton et al. [HSIW96] also generate a volumetric implicit-surface representation from a number of range images. Their method uses a local search to find the surface points from each view which are closest to a given voxel's center. Heuristics are then used to determine which of these closest points to use in estimating the signed distance. The main problem with their method is that their rules for determining which points to use in the average do not guarantee that the points are all part of the same portion of the object's surface, even in the case that the data is noise-free. The result is errors in the implicit surface estimate.

None of the three methods described above compensate for noise or extraneous point data—the data is assumed to be part of the object and noise is assumed to be negligible. Each of these methods suffer from inaccuracy due to their integration strategy (e.g., integrating unrelated observations). These accuracy problems will affect the result even when the data is perfect (i.e., noise-free).

2 Approach

To build a 3D surface model from multiple range images, we must address the following problems:

- View alignment: To begin to merge the data, the data from all the images must be in the same coordinate system.
- Data merging: We need to merge all the image data while eliminating or greatly reducing the effects of noise and extraneous data.

Several range images of an object are taken. Each range image consists of a set of 3D points in the coordinate system of the range sensor. The range image points are triangulated to a form a 3D surface mesh[1] using a standard threshold based technique to link neighboring image points [HSIW96, Whe96]. Our goal is to merge all these surfaces into a single model. The problem is that each view is taken from a different coordinate system with respect to the object. In order to compare or match the data from different views, we must transform all the data into the same coordinate system with respect to the object.

There are several ways we can approach this alignment problem—each requiring varying levels of human interaction: manual alignment, semi-automatic alignment, and automatic alignment. As view alignment is not the central focus of this work, we use controlled motion with calibration as it is currently the most practical option for an automatic solution. In our experimental setup, we calibrate two axes of a Unimation Puma robot with respect to a range-sensor coordinate frame. We can then mount the object on the robot's end effector and acquire images of an object at arbitrary orientations with known positions.

From this point we assume that the triangulated surfaces are aligned. Next, we consider the problem that we focus on in this paper: merging the data from aligned range images into a single model of the object's surface.

3 Data Merging

Given a number of triangle sets (surface meshes) which are aligned with respect to the desired coordinate system, we are now faced with the task of taking many triangulated surfaces in 3D space and converting them to a triangle patch surface model.

Even if we are given perfect sets of triangulated surfaces from each view which are perfectly aligned, the merging problem is difficult. It is difficult to determine how to connect triangles from different surfaces without knowing the actual surface beforehand. There are an exponential number of ways to connect two triangulated surfaces together—very few of which are acceptable.

The merging problem is exacerbated by noise in the data and errors in the alignment. Not only does the determination of connectedness become more difficult, but now the algorithm must also consider how to eliminate the noise and small alignment errors from the resulting model. Our solution makes use of a volumetric representation to avoid the difficulties associated with topology. This volumetric representation is described next.

3.1 Volumetric Modeling and Marching Cubes

Recently, the marching-cubes algorithm [LC87] has propelled volumetric modeling beyond the confines of "blocky" occupancy grids. Instead of storing a binary value in each voxel to indicate if the voxel is empty or filled, the marching-cubes algorithm requires that the data in the volume grid are samples of an implicit surface. In each voxel, we store the signed distance, $f(\mathbf{x})$, from the center point of the voxel, $\mathbf{x}$, to the closest point on the object's surface. The sign indicates whether the point is outside, $f(\mathbf{x}) > 0$, or inside, $f(\mathbf{x}) < 0$, the object's surface, while $f(\mathbf{x}) = 0$ indicates that $\mathbf{x}$ lies on the surface of the object.

[1] The need for using surface meshes rather than point sets is that we are interested in measuring the distance to the surface of the observed object. Doing so accurately requires that a surface be interpolated between the observed range image points. See [Whe96] for details.

The marching-cubes algorithm constructs a surface mesh by "marching" around the cubes while following the zero crossings of the implicit surface $f(\mathbf{x}) = 0$. The resulting surfaces are relatively smooth and their accuracy can be greater than the resolution of the volume grid due to sub-voxel interpolation.

The marching-cubes algorithm and the volumetric implicit-surface representation provide an attractive alternative to other conceivable mesh-merging schemes. First, they eliminate the global topology problem—how the various surfaces are connected—for merging views. The representation can model objects of arbitrary topology as long as the grid sampling is fine enough to capture the topology. Most importantly, the problem of creating the volumetric representation can be reduced to a single, simple question:

> *What is the signed distance between a given point and the surface?*

The given point is typically the center of a voxel, but we don't really care. If we can answer the question for an arbitrary point, then we can pose that same question at each voxel in the volume.

Now we may focus on a more easily defined problem: How do we compute $f(\mathbf{x})$? The real problem underlying our simple question is that we do not have a surface; we have many surfaces, and some elements of those surfaces do not belong to the object of interest but rather are artifacts of the image acquisition process or background surfaces. In the next subsection, we present an algorithm that answers the question and does so reliably in spite of the existence of noisy and extraneous surfaces in our data.

3.2 Consensus-Surface Algorithm

In this section, we present our answer to the question of how to compute the signed distance function $f(\mathbf{x})$ for arbitrary points $\mathbf{x}$ when given N triangulated surface patches from various views of the object surface. We call our algorithm the *consensus-surface algorithm.*

We can break down the computation of $f(\mathbf{x})$ into two steps:

- Compute the magnitude: compute the distance, $\mid f(\mathbf{x}) \mid$, to the nearest object surface from $\mathbf{x}$
- Compute the sign: determine whether the point is inside or outside of the object

We are given N triangle sets T_1,...,T_N—where T_i denotes the triangle set from the ith range image—which are aligned in the same coordinate system. The union of all triangle sets is denoted by $\mathbf{T} = \bigcup_i T_i$. Each triangle set, T_i, consists of some number n_i of triangles which are denoted by $\tau_{i,j}$, where $j = 1, ..., n_i$.

If the input data were perfect (i.e., free of any noise or alignment errors in the triangle sets from each view), then we could apply the following *naive algorithm*, Algorithm *ClosestSignedDistance*, to compute $f(\mathbf{x})$:

Algorithm *ClosestSignedDistance*
Input: point $\mathbf{x}$
Input: triangle set $\mathbf{T}$
Output: the signed distance d
1. $\langle \mathbf{p}, \hat{\mathbf{n}} \rangle \leftarrow$ *ClosestSurface*$(\mathbf{x}, T)$
2. $d \leftarrow \| \mathbf{x} - \mathbf{p} \|$
3. **if** $(\hat{\mathbf{n}} \cdot (\mathbf{x} - \mathbf{p}) < 0)$
4. **then** $d \leftarrow -d$
5. **return** d

where Algorithm *ClosestSurface* returns the point, $\mathbf{p}$, and its normal, $\hat{\mathbf{n}}$, such that $\mathbf{p}$ is the closest point to $\mathbf{x}$ from all points on triangles in the triangle set $\mathbf{T}$.

The naive algorithm for $f(\mathbf{x})$ finds the nearest triangle from all views and uses the distance to that triangle as the magnitude $\mid f(\mathbf{x}) \mid$. The normal of the triangle can be used to determine whether $\mathbf{x}$ is inside or outside the surface. If the normal of the closest surface point is directed towards $\mathbf{x}$, then $\mathbf{x}$ must be outside the object surface.

Again, the naive algorithm will work for perfect data. However, we must consider what happens when we try this on real data. The first artifact of real sensing and small alignment errors is that we no longer have a single surface, but several noisy samples of a surface (see Figure 3 in the results section for an example of the type of noise that may be present). Clearly, choosing the nearest triangle (as in Algorithm *ClosestSignedDistance*) will give a result as noisy as the constituent surface data. For example, a single noisy bump from one view can result in a bump on the final model.

A more sinister problem for the naive algorithm applied to real images is the existence of large errors and extraneous data. For example, it is not uncommon to see triangles sticking out of a surface or other triangles that do not belong to the object. This can occur due to sensor noise, quantization, specularities, interreflections and other possibly systematic problems of range imaging. This makes it very easy to infer the incorrect distance and more critically the incorrect sign. One badly oriented triangle can result in a voxel which has an incorrect sign for $f(\mathbf{x})$. The result will be in a hole rising out of the surface produced by the marching-cubes algorithm.

Our solution to these problems is to estimate the surface locally by averaging the observations of the *same surface.* The trick is to specify a method for identifying and collecting all observations of the same surface.

Nearby observations are compared using their location and surface normal. If the location and normal are within a predefined error tolerance (determined empirically), we can consider them to be observations of the same surface. Given a point on one of the observed triangle surfaces, we can search that region of 3D space for other nearby observations from other views which are potentially observations of the same surface. These searches are efficiently implemented using k-d trees [FBF77].

If an insufficient number of observations of a given surface are found, then the observation can be discarded as isolated/untrusted and the search can continue. Thus, we are requiring a quorum of observations before using them to build our model. The quo-

rum of observations can then be averaged to produce a *consensus surface*. This process virtually eliminates the problems described previously (with respect to the naive algorithm).

As an improvement over using an equally weighted voting scheme, we can assign a confidence value ω to each input surface triangle: weighting the surface points/triangles from a range image by the cosine of the angle between the viewing direction and the surface normal. This approach was used by Turk and Levoy [TL94] in their surface-mesh "zippering" algorithm. The confidence is simply computed by

$$\omega = \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}$$

where $\hat{\mathbf{v}}$ is the direction vector from the range sensor center of projection to the point on the triangle and $\hat{\mathbf{n}}$ is the normal of the given triangle. The consensus can now be measured as a sum of confidence measures and the quorum is over this weighted sum. The rationale is that two low-confidence observations should not have the same impact on the result as two high-confidence observations.

We can now specify the consensus-surface algorithm for computing $f(\mathbf{x})$:

Algorithm *ConsensusSignedDistance*
Input: point $\mathbf{x}$
Input: triangle set $\mathbf{T} = \bigcup_i T_i$
Output: the signed distance d
1. $\langle \mathbf{p}, \hat{\mathbf{n}} \rangle \leftarrow$ *ClosestConsensusSurface*$(\mathbf{x}, T)$
2. $d \leftarrow \| \mathbf{x} - \mathbf{p} \|$
3. **if** $(\hat{\mathbf{n}} \cdot (\mathbf{x} - \mathbf{p}) < 0)$
4. **then** $d \leftarrow -d$
5. **return** d

The only change from Algorithm *ClosestSignedDistance* (the naive algorithm) is that Algorithm *ConsensusSignedDistance* computes the closest consensus-surface point and its normal in line 1. The algorithm for computing the closest consensus-surface point and its normal is as follows:

Algorithm *ClosestConsensusSurface*
Input: point $\mathbf{x}$
Input: triangle sets T_i, $i = 1..N$, $\mathbf{T} = \bigcup_i T_i$
Output: the point and normal vector pair $\langle \mathbf{p}, \hat{\mathbf{n}} \rangle$
1. $O_{set} \leftarrow \emptyset$: the set of non-consensus neighbors
2. $C_{set} \leftarrow \emptyset$: the set of consensus neighbors
3. **for** each triangle set T_i
4. **do** $\langle \mathbf{p}, \hat{\mathbf{n}} \rangle \leftarrow$ *ClosestSurface*$(\mathbf{x}, T_i)$
5. $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle \leftarrow$ *ConsensusSurface*$(\mathbf{p}, \hat{\mathbf{n}}, T)$
6. **if** $\omega \geq \theta_{quorum}$
7. **then** $C_{set} \leftarrow C_{set} \cup \langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$
8. **else** $O_{set} \leftarrow O_{set} \cup \langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$
9. **if** $C_{set} \neq \emptyset$
10. **then** $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle \leftarrow \arg\min_{\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle \in C_{set}} \| \mathbf{x} - \mathbf{p} \|$
11. **else** $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle \leftarrow \arg\max_{\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle \in O_{set}} \omega$
12. **return** $\langle \mathbf{p}, \hat{\mathbf{n}} \rangle$

Algorithm *ClosestConsensusSurface* examines the closest point in each image's triangle set and searches for its consensus surface if one exists. After computing the closest consensus surface for each image, it chooses the closest of those from the consensus set C_{set}. C_{set} contains those locally averaged surfaces whose observations' confidence values sum to at least θ_{quorum} . Note that two consensus surfaces are not differentiated based on their confidence sum ω but rather on their proximity to $\mathbf{x}$, i.e., every consensus surface is considered to be a valid surface observation. If no consensus surfaces are found, the algorithm selects the average surface which has the highest summed confidence out of set O_{set}.

For completeness, we outline Algorithm *ConsensusSurface* which is required by line 5 of Algorithm *ClosestConsensusSurface*. Algorithm *ConsensusSurface* basically finds all surface observations that are sufficiently similar to the given point and normal. These observations are then averaged to generate a consensus surface for the input surface. This algorithm relies on the predicate

$$\text{SameSurface}(\langle \mathbf{p}_0, \hat{\mathbf{n}}_0 \rangle, \langle \mathbf{p}_1, \hat{\mathbf{n}}_1 \rangle) = \begin{cases} \text{True} & (\| \mathbf{p}_0 - \mathbf{p}_1 \| \leq \delta_d) \wedge (\hat{\mathbf{n}}_0 \cdot \hat{\mathbf{n}}_1 \geq \cos\theta_n) \\ \text{False} & \text{otherwise} \end{cases} \quad (1)$$

which determines whether two surface observations are sufficiently close in terms of location and normal direction, where δ_d is the maximum allowed distance and θ_n is the maximum allowed difference in normal directions. Now we present the pseudo code for Algorithm *ConsensusSurface*:

Algorithm *ConsensusSurface*
Input: point $\mathbf{x}$
Input: normal $\hat{\mathbf{v}}$
Input: triangle set $\mathbf{T} = \bigcup_i T_i$
Output: the consensus-surface point, normal, and confidence $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$
1. $\mathbf{p} \leftarrow \mathbf{n} \leftarrow \omega \leftarrow 0$
2. **for** $T_i \subset \mathbf{T}$
3. **do** $\langle \mathbf{p}', \hat{\mathbf{n}}', \omega' \rangle \leftarrow$ *ClosestSurface*$(\mathbf{x}, T_i)$
4. **if** SameSurface$(\langle \mathbf{x}, \hat{\mathbf{v}} \rangle, \langle \mathbf{p}', \hat{\mathbf{n}}' \rangle)$
5. **then** $\mathbf{p} \leftarrow \mathbf{p} + \omega' \mathbf{p}'$
6. $\mathbf{n} \leftarrow \mathbf{n} + \omega' \hat{\mathbf{n}}'$
7. $\omega \leftarrow \omega + \omega'$
8. $\mathbf{p} \leftarrow \frac{1}{\omega} \mathbf{p}$
9. $\hat{\mathbf{n}} \leftarrow \frac{\mathbf{n}}{\|\mathbf{n}\|}$
10. **return** $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$

Note that in Algorithm *ConsensusSurface*, the definition of Algorithm *ClosestSurface* was slightly modified to also return the confidence ω' of the closest surface triangle.

We refer to our algorithm as a whole as the consensus-surface algorithm. The following conditions are assumed:

1. Each part of the surface is covered by a number of observations whose confidences add up to more than θ_{quorum}.

2. No set of false surfaces with a sufficient summed confidence will coincidentally be found to be similar (following the definition of Equation 1) or this occurrence is sufficiently unlikely.

3. Given N surface views, the real surface is closest to $\mathbf{x}$ in at least one view.

If these assumptions are violated, mistakes in the surface mesh will result. From our experiments, a quorum requirement, θ_{quorum}, of 1.5 to 3.0 is usually sufficient to eliminate errors given a reasonable number of views of the object. While $\theta_{quorum} > 1$ implies that more than one view of each surface patch is available, this is only necessary if the image data warrants it. If no large errors exist, setting $\theta_{quorum} = 0$ will effectively do the job, returning simply the closest consensus surface (which may happen to be a single surface observation) and this would be the desired result. If any observations are integrated, the consensus surface algorithm ensures that the integration is done locally with respect to the closest surface point.

3.3 Accuracy and Efficiency

Volumetric modeling involves a tradeoff between accuracy and efficiency. The octree representation [Mea80] balances these problems while keeping the algorithm implementation simple. Instead of iterating over all elements of the voxel grid, we can apply a recursive algorithm on an octree that samples the volume more finely only when near the surface of the object. Since the surfaces of 3D objects are, in general, 2D manifolds in a 3D space, the octree in practice reduces the $O(n^3)$ storage and computation requirement to $O(n^2)$, where n is the number of samples along one dimension of the volume at the finest resolution.

To get a rough estimate of the cost of our model-building algorithm, let us assume that there are N views being merged, each triangle set T_i has n triangles, and that an $M \times M \times M$ voxel grid is used. The modeling algorithm will have expected cost $O(M^3 N^2 \log n)$. However, if octrees are used we may loosely assume that the number of voxels or octree elements which are evaluated will be proportional to the surface area of the object, reducing the expected cost to $O(M^2 N^2 \log n)$.

4 Experimental Results

Here we present some experimental results of our implementation of the 3D object modeling algorithm described in this paper. For our experiments, we selected a variety of objects to model using our system. We assume that the objects are rigid and opaque. Space limitations prevent us from describing all of our experiments; see [Whe96] for a complete description of our experiments. Here we describe an experiment of modeling a toy duck. In this example, specularities and interreflectance result in rather noticeable errors in the range data—presenting an especially difficult modeling problem.

For the example presented here we used 48 range images, each containing 256×240 pixels with each pixel containing a 3D coordinate. The resolution of data is approximately 1 mm, and the accuracy is on the order of 0.5 mm.

Figure 1 show the results, including an intensity image of the object, a close-up of some of the triangulated range images used as input to our algorithm (shaded to better indicate the roughness of the original data), a slice of the volume grid where the grey-scale indicates the proximity to a surface point (black closest, white furthest). Figure 2(a) shows three views of the resulting triangulated model. For this example, the 48 input images contained 555,000 triangles and the resulting model contained 27,000. The finest resolution of the voxel grid was 1.8 mm and approximately 4% of the $128 \times 128 \times 128$ volume grid was expanded by the octree. Parameters used were: $\theta_{quorum} = 2.25$, $\delta_d = 3$, and $\theta_n = 45$. Computing time was 52 minutes on an SGI Indy 5000 workstation.

As an example of what the naive algorithm, Algorithm *ClosestSignedDistance* of Section 3.2, would produce we show the example of the result of the naive algorithm on the duck data set in Figure 2(b). Notice how many extraneous surfaces exist near the duck from the input range-image data. The naive algorithm fails because it trusts that every surface observation is an accurate observation of the object surface. As can be seen from the sample range data of the duck in Figure 1, this is not the case.

To more clearly illustrate the accuracy of our modeling algorithm, Figure 3 shows a cross section of the final model and the original input range-image data. This example demonstrates the ability of our consensus-surface algorithm to accurately locate the surfaces in very noisy data. Figure 4 shows two other examples of objects (a piece of fruit and a mug) constructed using our algorithm. These and other experiments described in detail in [Whe96] demonstrate the ability of our algorithm to accurately model fine details and complicated geometries from very noisy input data.

5 Conclusion

We have described a method to create a triangulated surface mesh from N range images. Robotic calibration is used to acquire images of the object under known transformations, allowing us to align the images into one coordinate frame reliably. Our method for merging the range data, once aligned, takes advantage of the volumetric implicit-surface representation and the marching-cubes algorithm to eliminate topological problems of stitching meshes together.

The main contribution of this paper is our robust algorithm for merging data from multiple views: the consensus-surface algorithm which attempts to answer the question

> *What is the closest "real" surface to a given point?*

With the answer to this question, we can easily compute the signed distance $f(\mathbf{x})$ correctly. While other known methods also address this question, their algorithms for computing $f(\mathbf{x})$ have accuracy problems even in cases where the input range data is perfect (i.e., noise-free). None of the methods explicitly deal with significant amounts of noise and errors.

In contrast, our algorithm attempts to justify the selection of observations used to produce the average by finding a quorum or consensus of locally coherent observations. This process eliminates many of

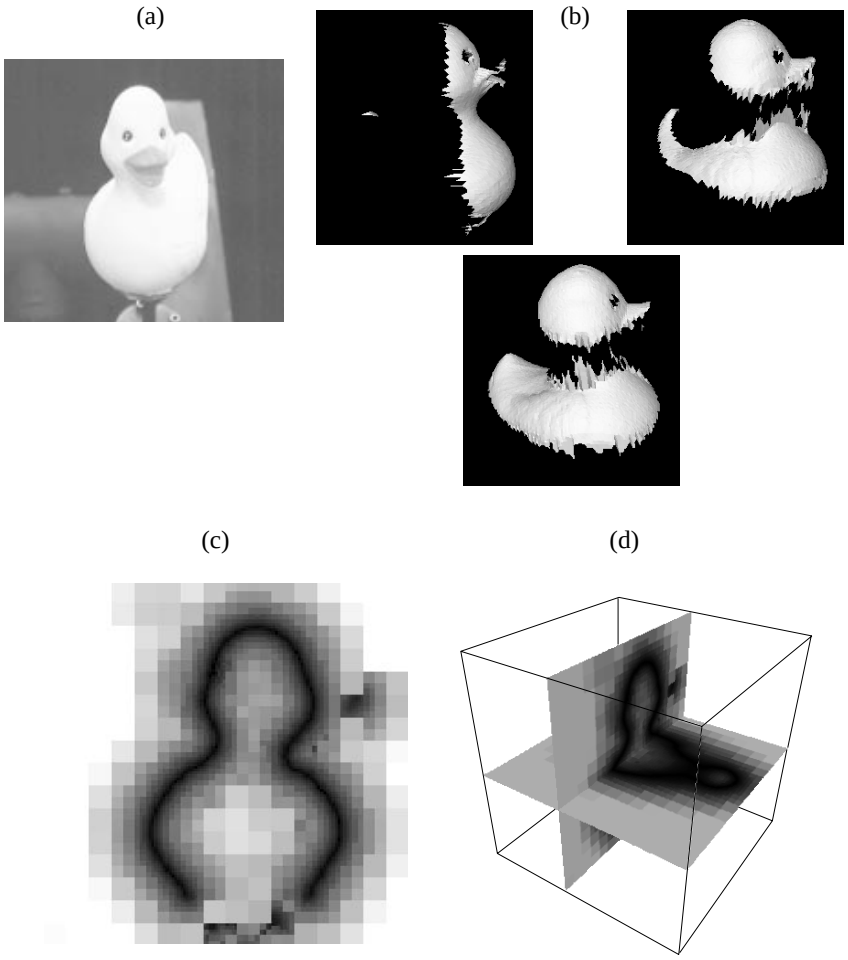

Figure 1: Results from modeling the rubber duck. (a) An intensity image of the duck, (b) a close-up of some of the triangulated range images used as input to the consensus-surface algorithm, (c) a slice of the resulting implicit-surface volume where darker points are closer to the surface, and (d) a 3D view of two cross sections of the implicit-surface octree volume.

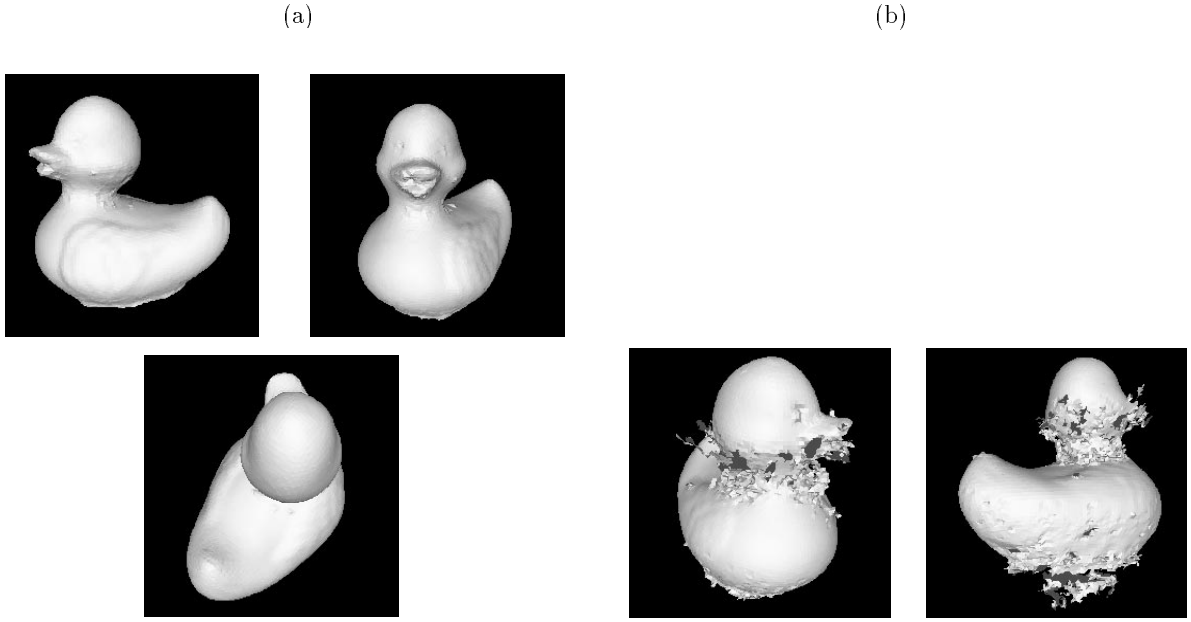

Figure 2: (a) Three views of the resulting triangulated surface model of the duck. (b) Two views of the surface model produced by the naive algorithm, Algorithm *ClosestSignedDistance*, using the same image data.

the troublesome effects of noise and extraneous surface observations in our data. Our algorithm will give the desired results when the data is noise-free as well.

Consensus surfaces can be computed independently for any point in the volume. This feature makes it very easy to parallelize and allows us to straightforwardly use the octree representation. The octree representation enables us to model objects with high accuracy with greatly reduced computation and memory requirements.

Our experimental results demonstrate that our consensus-surface algorithm can construct accurate geometric models from rather noisy input range data and somewhat imperfect alignment.

Acknowledgments

The authors thank Heung-Yeung Shum, Peter Rander, Andrew Johnson and Martial Hebert for many helpful conversations and help with the experimental setup and implementation.

References

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*, pages 303–312. ACM, 1996.

[FBF77] Jerome H. Friedman, Jon Bentley, and Raphael Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J.A. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Proceedings of SIGGRAPH*, 26(2):71–78, 1992.

[HSIW96] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Proceedings of the European Conference on Computer Vision*, pages 117–126. Springer-Verlag, 1996.

[LC87] W.E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH*, pages 163–169. ACM, 1987.

[Mea80] D. J. R. Meagher. *The octree encoding method for efficient solid modeling.* PhD thesis, Rensselaer Polytechnic Institute, 1980.

[TL94] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH*, pages 311–318. ACM, 1994.

[Whe96] Mark D. Wheeler. *Automatic Modeling and Localization for Object Recognition.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1996.

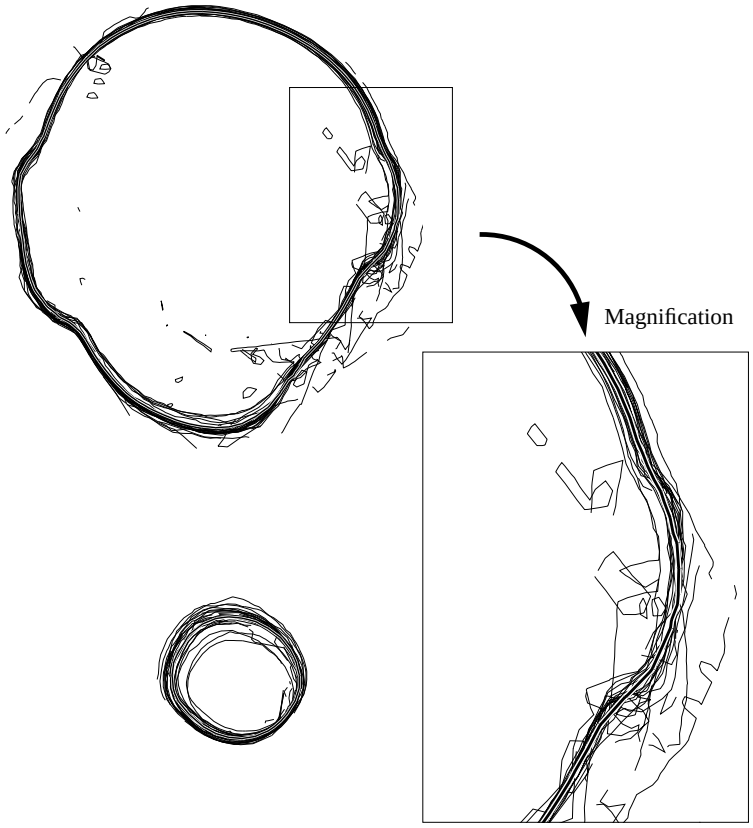

Figure 3: A cross section of the final model of the rubber duck (thick black line) and the original range-image data (thin black lines) used to construct it.

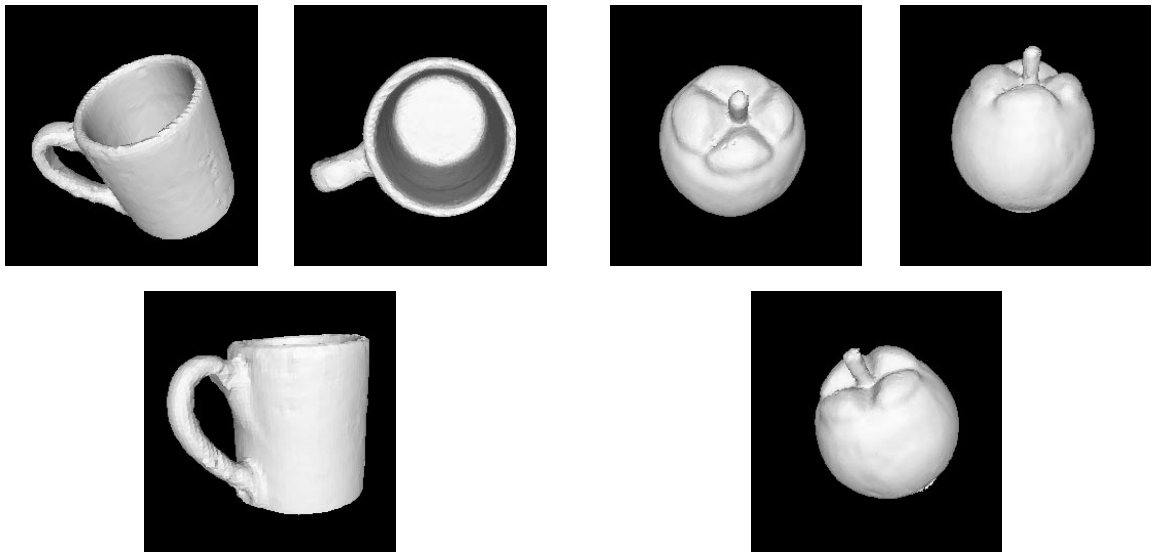

Figure 4: A cross section of the final model of the rubber duck (thick black line) and the original range-image data (thin black lines) used to construct it.